Evaluating Cloud Migration Options for Relational Databases

Martyn Holland Ellison

Doctor of Engineering

University of York
Computer Science

September 2017

Abstract

Migrating the database layer remains a key challenge when moving a software system to a new cloud provider. The database is often very large, poorly documented, and used to store business-critical information. Most cloud providers offer a variety of services for hosting databases and the most suitable choice depends on the database size, workload, performance requirements, cost, and future business plans. Current approaches do not support this decision-making process, leading to errors and inaccurate comparisons between database migration options. The heterogeneity of databases and clouds means organisations often have to develop their own ad-hoc process to compare the suitability of cloud services for their system. This is time consuming, error prone, and costly.

This thesis contributes to addressing these issues by introducing a three-phase methodology for evaluating cloud database migration options. The first phase defines the planning activities, such as, considering downtime tolerance, existing infrastructure, and information sources. The second phase is a novel method for modelling the structure and the workload of the database being migrated. This addresses database heterogeneity by using a multi-dialect SQL grammar and annotated text-to-model transformations. The final phase consumes the models from the second and uses discrete-event simulation to predict migration cost, data transfer duration, and cloud running costs. This involved the extension of the existing CloudSim framework to simulate the data transfer to a new cloud database.

An extensive evaluation was performed to assess the effectiveness of each phase of the methodology and of the tools developed to automate their main steps. The modelling phase was applied to 15 real-world systems, and compared to the leading approach there was a substantial improvement in: performance, model completeness, extensibility, and SQL support. The complete methodology was applied to four migrations of two real-world systems. The results from this showed that the methodology provided significantly improved accuracy over existing approaches.

Contents

List of Figures

List of Tables

Author Declaration

I declare that this thesis is a presentation of original work and I am its sole author. This work has not previously been presented for an award at this, or any other, University. All sources are acknowledged as References. Parts of this thesis have been published previously in the following research papers:

- M. Ellison, R. Calinescu, and R. F. Paige, "Towards Platform Independent Database Modelling in Enterprise Systems," in *5th International Symposium on Data to Models and Back (DataMod)*, Federation of International Conferences on Software Technologies: Applications and Foundations, pp. 42–50, Springer, 2016
- M. Ellison, R. Calinescu, and R. F. Paige, "Re-engineering the Database Layer of Legacy Applications for Scalable Cloud Deployment," in *7th International Conference on Utility and Cloud Computing (Doctoral Symposium)*, pp. 976–979, IEEE, 2014
- M. Ellison, R. Calinescu, and R. F. Paige, "Evaluating Cloud Database Migration Options Using Workload Models," *Journal of Cloud Computing Advances, Systems and Applications*, vol. 7, Mar 2018

The algorithms, tools, and research data from this thesis have been published on-line:

- M. Ellison, "DBLModeller Tool." `http://mhe504.github.io/DBLModeller`, Jul 2017
- M. Ellison, "MigSim Tool." `http://mhe504.github.io/MigSim`, Jul 2017
- M. Ellison, "Option Evaluation Methodology (Eclipse Process Framework)." `https://mhe504.github.io/Migration-Option-Evaluation`, Aug 2017

Chapter 1

Introduction

Cloud computing is one of the leading paradigms for hosting a software system or service. There are several advantages compared to using an in-house datacenter [12], most notably: the flexibility to closely match resource demands with supply, reduced configuration effort [13], and reduced maintenance effort [14]. This ability to quickly reconfigure a system's infrastructure to match its load (and meet service level agreements) typically reduces costs [15]. Furthermore, many configuration tasks like updating software, configuring database clusters, or replacing failed hardware are performed by the cloud provider.

The advantages of cloud computing mean many organisations are designing their new systems to be cloud-native [16] [17]. They may have migrated their existing systems to the cloud, or are considering doig so [18]. An organisation may also want to move between cloud providers to avail of new features or reduced costs. However, migrating a system to a new platform is a challenging, complex, and high-risk project [19] [20].

Many of the tasks involved in the migration (e.g., identifying incompatibilities [21] or moving big data [22]) are supported by existing methods and tools. These have been developed by the public cloud providers and the academic research community. However, the complex nature of software systems and the continuous evolution of cloud services means numerous challenges remain. This thesis focuses on the challenges associated with the database layer, which is typically the largest and most valuable component of a system.

In this chapter the field of cloud migration is introduced, along with the research gap which this thesis will address. Next, four key research problems are explained. Finally, the thesis structure is presented and the content of each chapter is briefly summarised.

1.1 Background and Motivation

Large enterprise systems are typically structured as several layers or components. For migration, the type of activities performed on each depends on whether they are categorised as *software* or *database*. In this context software is a component providing a business function, user interface, or linking together other components or layers. Migrating the software layer of a system to the cloud usually requires: selection of the most suitable cloud provider, choosing the specification of virtual machines, and configuration changes. It may also need to be re-engineered to run on a particular cloud platform or to exploit new cloud services. Approaches and tools created by public cloud providers [23] [24] and independent researchers [25] [26] support all these tasks at least to some extent.

Often the database layer is the most challenging aspect of a cloud migration, primarily due to the size and importance of the data [27]. The above tasks for the software layer also apply to the database layer, although several new tasks are also required. An organisation must estimate the data transfer duration to choose the best approach (e.g., over the Internet or device shipping [22]). They must decide if system downtime is acceptable and plan accordingly. Finally, they need to estimate the cost of the migration and include this in the project budget (a zero-downtime migration will increase cost). All of these tasks require a detailed understanding of the database size, structure, and workload; which the organisation rarely has with a legacy system.

The heterogeneity of software systems, databases, and cloud providers is a major obstacle when developing an approach for migration [28]. Model-Driven Engineering principles are typically used to address this. A model (i.e., a description of some system artefact) is obtained and treated as a first-class entity in the migration process. Any required re-engineering is performed by executing transformations on the model, first to refine it, then to generate source code. Furthermore, system models can be compared against models of cloud services to determine: required changes, suitability, and costs.

The existing methodologies and approaches support the selection of cloud providers, services, and instance types only for the software layer [21]. The existing approaches for data migration and database modernisation do not consider cloud-specific concerns (e.g., pay-per-use pricing and the implications of using a 'database-as-a-service') [29]. The gap between the support available for migrating the software and database layers can cause organisations to make database migration decisions based on inaccurate estimates, or define ad-hoc processes as they go.

1.2 Research Problems and Contributions

From the numerous challenges of migrating a relational database to a new cloud platform, this research focuses on understanding the database, and providing the information necessary to choose a migration approach and cloud platform. Specifically, the following high-level research problems will be addressed:

RP1 No systematic method of estimating the duration of a cloud database migration exists, which is essential when selecting a migration approach and in planning its execution.

RP2 The available approaches for estimating cloud database running costs for a system rely on the user knowing future resource requirements, leading to inaccuracies.

RP3 Database modelling approaches, suitable for use in model-based migration or modernisation, are vendor-specific and target a narrow range of databases.

RP4 These approaches do not consider the cloud-relevant properties of the database structure and workload.

The migration duration (i.e., the time taken to move the data to a new platform) must be known to choose the most suitable migration method [30]. Furthermore, an accurate estimation of future annual cloud running costs allows budgeting and comparison. These costs are not fixed: when the organisation and the system load grow, additional cloud resources may be needed. Existing cloud cost calculators [31, 32] or approaches [33, 34] provide limited support for estimating these costs. They require an organisation to accurately know the resources they require, which is a complex task. They also fail to support expected growth trends.

A database modelling approach is deemed suitable for cloud database migration if it considers (at least): definitions of data types, relationships between elements, table sizes, load type and pattern, and growth trends. These desirable properties have been identified from existing literature on model-based migration and modernisation, as presented in Chapter 2. Together they affect the required type and quantity of cloud database instances, and therefore the cost of migrating the database to, and running it on, the cloud.

The issue of vendor-specific approaches includes the target metamodel as well as the source database. Existing software-layer migration and modernisation approaches [26] [25] have converged on a standard metamodel — the Knowledge Discovery Metamodel

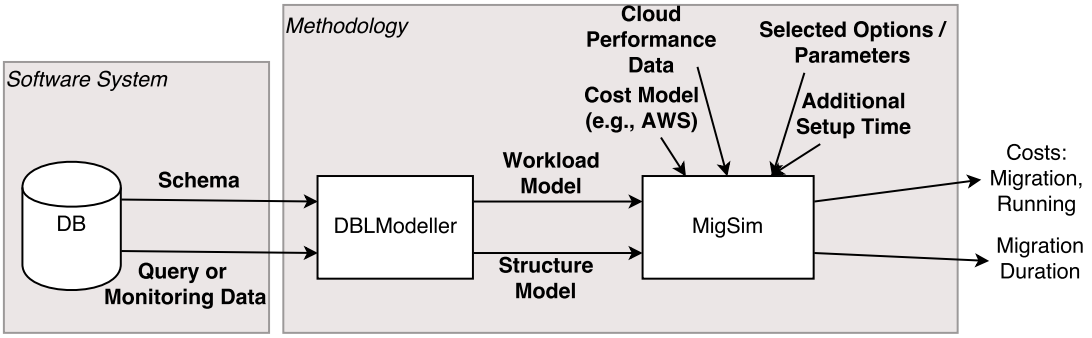

Figure 1.1: The proposed solution including the key approaches: DBLModeller and MigSIM

[35] — which also supports the database-layer. However, as discussed in Chapter 2, a range of different metamodels have been proposed and used in the database layer modelling community. This makes tool interoperability and integration challenging.

These research problems are addressed in the thesis with tool-supported approaches for: database modelling (called DBLModeller) and migration simulation (called MigSim). An end-to-end methodology using DBLModeller and MigSim has also been created. This enables estimation of the time required for data migration, migration cost, and future running costs. These estimates are based upon the workload and size of the database, the target cloud charges, and the performance of the selected cloud instances. Each parameter can be adjusted to evaluate different cloud database migration options. The methodology allows organisations to ask questions, such as: 'Would the cost of increasing the database bandwidth to speed-up migration decrease overall project costs?' or 'Will the database need to be scaled-up within the current financial year if growth trends continue?'.

The relationship between DBLModeller, MigSim, and the new methodology is illustrated in Figure 1.2. First, the user must obtain a SQL schema and workload data from the database within their existing software system. The workload data can be obtained from query logs or existing monitoring tools. Next, they use DBLModeller to extract standardised workload and structure models from this input data. Finally, they run MigSim with the models to produce the cost and duration estimates. MigSim also requires: (1) a cloud model which describes the charges levied by the desired cloud provider, (2) performance data for the database instance in the simulation, (3) the set of migration options being evaluated, and (4) an 'additional time' estimate. The 'Additional time' is the duration the infrastructure will be idle for non-migration tasks,

such as setup.

The industrial sponsor for this Engineering Doctorate was Science Warehouse, [1] who are a UK-based company specialising in procurement services. They develop a set of web applications in-house, which support: catalogue-based procurement, analysis of expenditure within an organisation, and invoicing. These are implemented in Java and use an Oracle database. Their long-term goal has been to migrate these applications (and various development and test environments) from physical servers to the cloud. The work presented in this thesis has been evaluated on Science Warehouse's applications along with open-source systems from different domains. Furthermore, the company has advised on the industrial relevance of this work.

Contributions

The contributions of this work cover three areas: database modelling, database migration simulation, and the evaluation of cloud database migration options. Together these contribute to reducing costs for organisations performing cloud database migration by increasing automation and accurately evaluating the impact of cloud deployment options.

1) Modelling

The *DBLModeller* approach and model extraction tool have been developed. This extracts database structure models that conform to the Knowledge Discovery Metamodel (a standard in model-based software modernisation [35]). Secondly, it extracts a database workload model which conforms to the Structured Metrics Metamodel [2] (another common standard). The approach defines where and how inputs to the tool should be collected.

The use of standards within Model-Driven Engineering supports re-use of models and tool interoperability, while also solidifying best practice [36]. Furthermore approaches built on standards are more likely to be adopted by industry [37]. Using the KDM and SMM strengths this work and contributes to the MDE community.

The DBLModeller tool implements some novel changes to *Grammar-to-Model* [38] model extraction: (1) Annotations, (2) a Multi-dialect grammar, and (3) Usage surveying. The impact of these changes on SQL support, tool extensibility, and performance has been evaluated in Section 3.3. A study of eighteen SQL schemas showed the DBLModeller tool supported between 96% and 99% of their content, representing a

[1] `http://www.science-warehouse.com`

significant improvement over the Grammar-to-Model tool. An extensibility case study using a Microsoft SQL Server schema showed DBLModeller required 71 modified lines of code, whereas Grammar-to-model required 206. Finally, a performance study using six schemas showed DBLModeller can extract a model in between 84% and 86% less time on the same hardware. These findings can be applied in-principle to other model extraction problems.

DBLModeller is the first platform-independent approach to extract KDM-based models from large SQL databases. Existing approaches (e.g., [39]) have focused on a single database platform with a small number of tables. It is also the first approach to extract SMM-based workload models. The KDM structure and SMM workload models provide a detailed understanding of a legacy system's database, which is necessary for cloud migration.

2) Migration Simulation

The MigSim discrete-event simulator is an extension of CloudSim [40]. It predicts migration costs, data transfer duration, and future running costs by simulating data transfer between system infrastructure components. Functionality has been added to CloudSim which creates objects in the simulation from KDM and SMM based models. This removes the need to write or modify code to simulate different systems, as was previously the case. KDM and SMM are commonly used metamodels within the field of model-driven software modernisation, so mappings between KDM-CloudSim and SMM-CloudSim are valuable in other situations (e.g., software layer migration simulation). No other cloud simulation tool can currently simulate database migration and operation.

A Cloud Cost Metamodel has been developed so that charges levied by public cloud providers can be input into MigSim. The metamodel provides a standard interchange format which could be used for price comparisons or within other model-based approaches. Many cloud metamodels exist already (identified in Section 2.2.3), although no clear standard has emerged and none focus exclusively on pricing. Extending or adapting these metamodels would be significantly less intuitive for users, hence making the models more challenging to produce.

The accuracy of the predictions produced by MigSim were evaluated using: (1) a migration case study and (2) a set of running cost experiments. In the migration case study (Section 4.4), databases from two large enterprise systems (Apache OFBiz and Science Warehouse) were migrated between the Microsoft Azure and AWS public clouds. The migration duration and actual cost incurred were compared against the

simulated results, showing a relative error of between 4% and 22%. The running cost experiments (Section 4.4.2) evaluated *auto-scaling* in MigSim, as this is the key factor determining the accuracy of the predicted future cloud running costs. This involved the deployment of Apache OFBiz onto Microsoft Azure and AWS cloud instances, synthetic load was then applied to both databases until they failed to process SQL operations. The load at this point was recorded and compared to the simulation, showing a relative error of between 8% and 12%.

3) Evaluation of Migration Options

The migration option evaluation methodology (presented in Chapter 5) defines how DBLModeller and MigSim can be used together to accurately evaluate costs and migration duration. It allows organisations to assess the impact of different migration and cloud infrastructure. The results produced by applying the methodology are shown to have a high level of accuracy, making them suitable for use in industry.

The methodology also defines how the inputs for DBLModeller and MigSim can be obtained. This includes identifying existing sources of data which describe the database workload, or obtaining it via query logging if necessary. Furthermore, it highlights the key migration options which should be considered when migrating a database to the cloud. The evaluation of DBLModeller and MigSim involved the modelling a migration of real-world databases at Science Warehouse (an industrial research partner). The tasks identified during this process are incorporated into the methodology.

In summary, the DBLModeller approach and the support it added for database-layer modelling with KDM/SMM addresses Research Problem 1. The database properties it supports — which were selected based on the literature and collaboration with Science Warehouse — addresses Research Problem 2. Research Problem 3 is addressed by the cloud database migration planning methodology. The MigSim discrete-event simulator addresses Research Problem 4 by enabling organisations to use database models to estimate cloud costs.

1.3 Thesis Structure

The remainder of the thesis is organised as follows. Chapter 2 introduces key concepts and terminology used throughout this thesis and reviews the related work on cloud migration. It includes sections on: model-based software modernisation (and how it supports the cloud), migration methodologies, migration case studies, and cloud simulation. Notable approaches include the REMICS [25] and ARTIST [26] method-

ologies for migrating legacy systems (on non-virtualised infrastructure) to the cloud. The CloudMIG [21] simulation tool, which assess a systems cloud compatibility via simulation, is also presented.

Chapter 3 presents the DBLModeller approach, tool implementation, and its evaluation. Each of the steps (input gathering, transformation, and refinement) in the approach are explained. This includes the algorithm and a time complexity analysis for the refinement step. An extensive evaluation is presented, where DBLModeller was applied to fifteen systems from a variety of domains. The characteristics of each system have been described so that comparisons can be drawn.

Chapter 4 presents the approach for simulating cloud migration and deployment of relational databases. This includes methods for cloud cost modelling and determining database instance performance (two necessary inputs in addition to the models from DBLModeller). A detailed design of the simulation is presented, along with an explanation of CloudSim extensions. Finally, the approach is evaluated by comparing the simulation results with four migrations of real-world databases and two cloud database capacity experiments.

Chapter 5 introduces the end-to-end methodology for evaluating cloud database migration options. This methodology combines the DBLModeller and MigSim approaches. The scope (i.e., supported options) is identified and justified. The methodology has been evaluated within the previous chapters (Section 3.3 and 4.4), although these results are analysed further and some specific threats to validity are considered.

Finally, Chapter 6 summarises contributions of the project and provides further insights on its research findings. In addition, areas of potential future work are outlined.

1.3.1 Intended Audience

The intended audience for this thesis includes: (1) organisations at the planning or feasibility assessment stage of a cloud migration, (2) organisations wanting detailed knowledge of their database workload, (3) model-driven engineering researchers, and (4) cloud migration researchers. The migration cost and duration estimates produced by the methodology in Chapter 5 will be of greatest value for medium to large datasets (e.g., 50GB to 1TB). The costs for smaller systems will be less important, while on-demand cloud instances are unlikely to be used for extremely large datasets. The cloud running cost estimates also produced by the methodology will be valuable for databases whose load is not primarily driven by unpredictable events.

The contributions made in Chapter 3 have applications beyond cloud migration

of databases, hence this work will likely be of interest to the wider MDE community. The contributions from Chapters 4 and 5 focus on cloud migration. They have relevance to researchers working with cloud migration methodologies, database migration algorithms, and cloud simulation.

Chapter 2

Literature Review

2.1 Introduction

This chapter reviews the literature in the cloud database migration field. It focuses on work related to the four research problems being addressed in this thesis. The strengths and weaknesses of the various approaches and tools have been identified.

Section 2.2 introduces key Model-Driven Engineering concepts, Architecture Driven Modernisation, Cloud Cost Modelling, and Grammar-to-Model mapping. Many of the contributions made in this research project are in the Architecture Driven Modernisation (ADM) field, and the metamodels developed by the Object Management Group ADM Task Force [41] are used extensively. The limitations of leading cloud cost modelling approaches are identified, forming a basis for the work in Section 4.3. The Grammar-to-Model mapping approach is examined in detail due to is suitability for addressing research problem RP3.

Section 2.3 presents cloud computing concepts, such as platform types and deployment models, which are referenced throughout the thesis. Next, the key work which examines cloud migration challenges is reviewed to highlight the novelty of the Cloud Database Migration Planning Methodology (Chapter 5). The ARTIST and REMICS European Union research projects developed cloud migration methodologies, they are both examined here to show the current state-of-the-art. Subsection 2.3.3 presents work on the problem of allocating appropriate cloud resources to existing systems, such as CloudMIG and Log2cloud, as this is an important consideration in MigSim (Chapter 4). Finally, major cloud migration case studies are reviewed to determine the industrial impact of this research project.

The process of migrating a system to the cloud has similarities to software mod-

ernisation and migrations between physical servers. As a result, Section 2.4 reviews key work on model-based refactoring of the software and database layers. MoDisco (Section 2.4.1) is a leading KDM model extraction tool which has the potential to be used alongside DBLModeller to produce complete models of a system. NoSQL and polyglot persistence (using multiple database types in a single system) are common database technologies and their impact on research problem RP3 is considered.

Section 2.5 presents the leading discrete-event simulators intended for cloud computing or large datacenters. These have several valuable features which could be used when addressing research problem RP2. Furthermore, relevant interpolation and extrapolation methods which could be applied to simulation data are introduced. The strengths and limitations of these methods are considered, alongside an example illustrating their use.

2.2 Model-Driven Engineering

Model-Driven Engineering (MDE) is a paradigm where models are first-class entities and used as the basis for the software development process [42]. This can include development of new systems and modernisation of existing systems. Modelling is a valuable tool to handle complexity, aid understanding, and increase automation. As a result it is used extensively in approaches for cloud database migration.

2.2.1 Concepts

The two core concepts of MDE are: models and transformations [43]. A model can be defined as a *description of a phenomena of interest* [44] in a textual or graphical notation. Models can exist at several levels of abstraction and are not necessarily a simplification (although they are often described as this [45]).

Modelling notations (designed specifically for this purpose) are referred to as metamodels. They specify models and define what can be considered valid [46]. Example graphical notations/metamodels include the Unified Modelling Language [47], while Domain-Specific Languages are often used for textual notions. A Domain-Specific Language is one tailored to a specific domain or modelling problem. It is based on the concepts, features, and terminology of that domain [48]. Models may also be expressed in a formal mathematical notation.

A *transformation* is an operation which produces a new model or artefact [42]. They are used to: translate models into different notations/metamodels, move between

levels of abstraction, refine models, and refactor models. Three common categories exist: Text-to-Model transformations (T2M), Model-to-Model (M2M), and Model-to-text (M2T) [43]. In model-driven software engineering text is typically source code, although it could be any artefact or piece of information.

2.2.2 Architecture-Driven Modernisation

Architecture-Driven Modernisation (ADM) refers to a set of tool-enabled disciplines on the analysis, re-factoring, and transformation of existing software [49]. In 2003 the OMG ADM task force was established to work on standards in this area. Their key aim was to standardise the information and meta-data that the tools generated, therefore aiding interoperability. The diverse nature of systems, platforms, and languages, means that a single tool will rarely be sufficient for a modernisation project and interoperability is essential.

Ulrich and Newcomb [49] describe modernisation workbenches as key tools within ADM. A modernisation workbench is a tool which provides a baseline set of modernisation functions, including: parsing tools to analyse the artefacts of the system, system workflow identification, program logic and structure analysis, re-factoring, and metric extraction. Re-factoring tools can support language and platform migration, code modularisation, and model extraction.

Two key metamodelling standards developed by the OMG are the Knowledge Discovery Metamodel (KDM) and the Abstract Syntax Tree Metamodel (ASTM), these are used extensively within the tools produced by the REMICS project (Section 2.3.2) and MoDisco (Section 2.4.1). The KDM — which is defined in ISO 19506 — provides a comprehensive view of "as is" application and data architectures. Artefacts of the existing software are represented in a platform and language independent manner, as entities, relationships, and attributes. The ASTM is designed to represent software at the level of procedural logic, data definition, and workflow composition. The ASTM standardises the format of Abstract Syntax Trees (ASTs) and defines a method for their representation.

Other notable standards include the Structured Patterns Metamodel (SPMS) [50] for defining architectural patterns and anti-patterns, the Structured Metrics Metamodel (SMM) [2], and 'ADM visualisation' which aims to deliver a consistent set of visualisations across tools [49].

Knowledge Discovery Metamodel

The Knowledge Discovery Metamodel (KDM) was developed by the Object Management Group to enable: (1) the storage of facts about an information system, (2) analysis and reasoning about facts, and (3) the exchange of facts in a common interchange format [35]. In this context a 'fact' is any piece of information on an *existing* software system. Unlike UML or similar software system modelling notations where the target system may or may not exist, KDM is designed to be used in a bottom-up process to model a system which is live/in-use. In such scenarios there is more information to be captured and modelled, which is a key challenge the developers faced. KDM has been defined in ISO/IEC 19506 [51] and an implementation has been provided in the Eclipse Modelling Framework [52]. Here, the models are stored as XML.

KDM is the cornerstone of Architecture-Driven Modernisation. It is used in the reverse engineering phase of this approach to provide one or more representations of the legacy system (at different abstraction levels). During the forward engineering phase it is used as a repository of information on all existing system artefacts (e.g., source code, databases, user interfaces, and business rules). Tools such as MoDisco [53] can extract KDM models from source code. While the CDOSim [54], REMICS [25], and ARTIST [55] cloud modernisation approach consume KDM models. These are explained later-on in this literature review.

The Knowledge Discovery Metamodel consists of layers which can represent physical and logical artefacts of a legacy system. These are (in order of abstraction from low to high): Infrastructure, Program Elements, Runtime Resources, and Abstractions Layer. Each layer contains multiple packages, as illustrated in Figure 2.1.

The Infrastructure layer contains the: Core, KDM, and Source packages. The core package contains the abstract/root elements of the metamodel, which all other elements must inherit. The KDM package builds on the Core package to provide other elements commonly used in the metamodel. While the Source package provides all the elements needed for an 'Inventory' model of the system (i.e., all its source files, image files, and other artefacts).

The Program Elements layer contains the Code and Action packages. Code contains elements to represent various common components in programming languages, such as: types, classes, procedures, methods, templates and interfaces. This is the lowest-level representation of a systems functionality. Action enables the representation control or data flow within the system.

The Runtime Resources layer contains the: Data, Event, UI, and Platform packages.

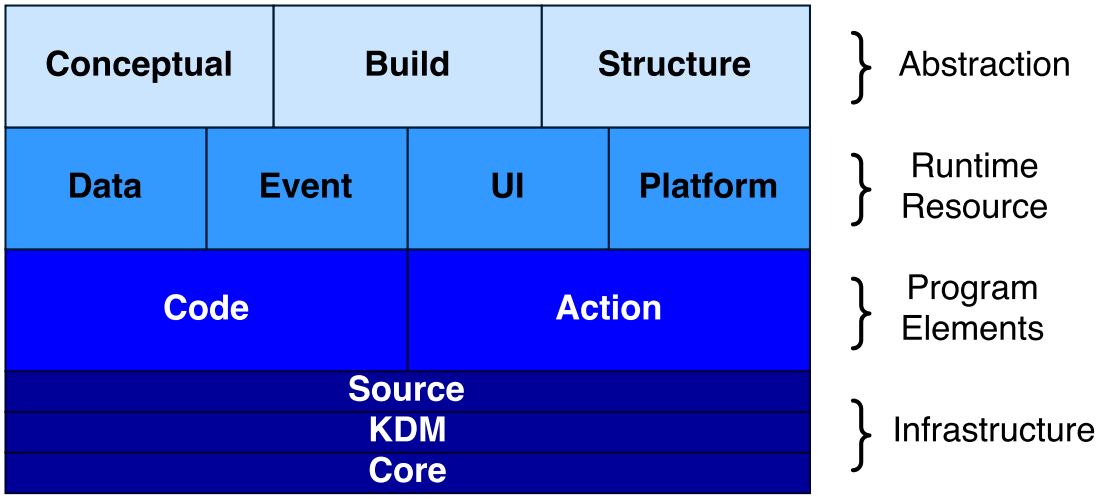

Figure 2.1: Layers and packages within the Knowledge Discovery Metamodel (adapted from Fig 8.1 in [1])

The data package can represent complex data repositories like: relational databases, record files, XML schemas, and XML documents. The event package can represent state and state transitions. The UI package models fields, buttons, events, and similar visual objects. Finally, the platform package represents the system's runtime operating environments.

The top-level Abstractions layer contains the following packages: Structure, Conceptual, and Build. This layer is intended to represent derived or implicit facts obtained from the lower layers. The structure package represents the logical system architecture/structure, and the conceptual package represents domain-specific information, such as terminology, concepts and roles. The build package represents the processes and activities involved in building/compiling the system.

As the Knowledge Discovery Metamodel is relatively large, its developers have introduced *compliance* levels. This allows tools to be categorised based on the section(s) of the metamodel they support. A tool is level 0 compliant when it fully supports: core, kdm, source, code, and action. Next, a level 1 tool adds supports for one or all of the: platform, data, event, UI, build, structure, and conceptual packages. Finally, level 2 tools support the entire metamodel.

Some criticisms of the KDM have been made in [56] and [57], where it is argued that the extension mechanism is too restrictive. This is backed up in [58] where a 'heavy-weight' extension was performed where the KDM was modified instead of using its light-weight extension mechanism. Another issue with the KDM is the limited tool

support. Most tools only support a small subsection of the metamodel, like [53], and do not claim support of any compliance level.

Structured Metrics Metamodel

The Structured Metrics Metamodel (SMM) [2] has two purposes: (1) to provide a standard format for exchange of measurements, and (2) to allow definition of measures. Combining the measurements and their definition in a single model enables understanding and tool support. The metamodel's developers (the Object Management group) argue that measurements are critical for many engineering decisions (e.g., checking if system requirements are met), and therefore any model-based software development approach should make use of it.

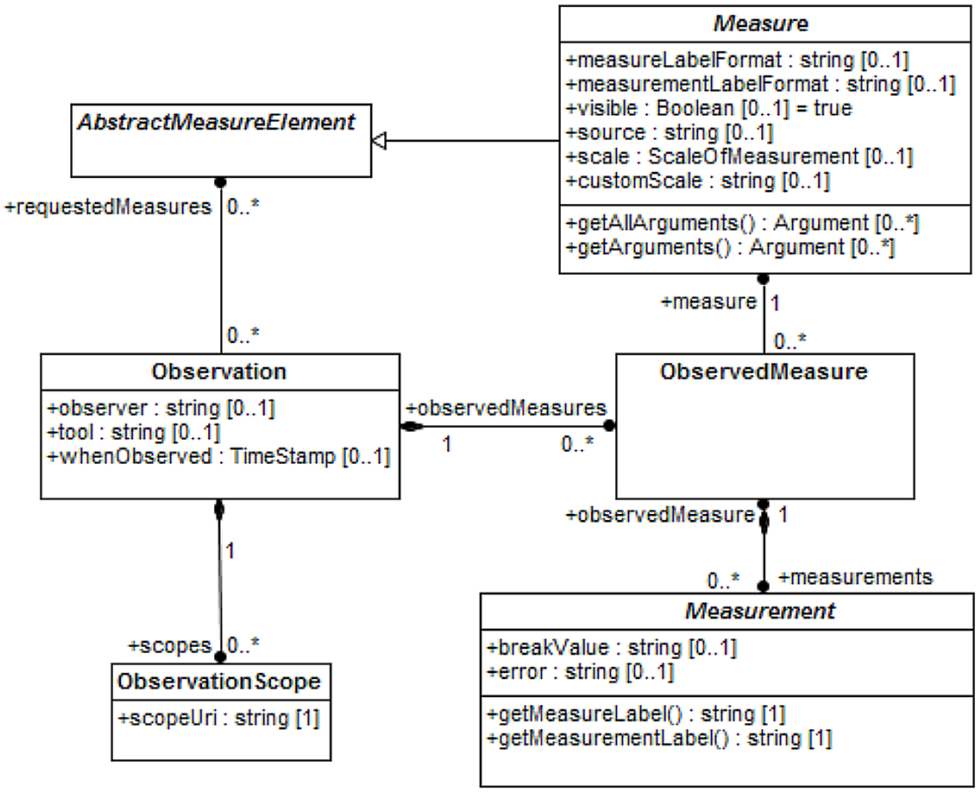

Figure 2.2: Key elements of the Structued Metrics Metamodel and their relationships (adapted from Fig 7.1 in [2])

The first step in creating a SMM instance is to define the measure 'library'. This should contain a set of measure definitions for the current project. For example, in a legacy system re-engineering project the library may contain code quality measures

like unit test coverage or complexity. The second step is to observe the system in a particular state and apply the measure, therefore creating a measurement. Each observation will have: a time, a scope (e.g., complexity of component A), and optionally a tool which performed it. The library is not intended to support the 'execution' of the measures on some model, but rather to explain measurements. Figure 2.2 is a fragment of the metamodel to illustrates the relationship between measures, observations and measurements.

Additionally, SMM supports direct and derived measures. Direct measures are created from the software system or artefact, whereas derived measures are calculated from direct measures. For example, a direct measure could be complexity of a component, and a derived measure could be the average complexity of all components.

A simple extension mechanism is included in the metamodel. Users can associate an annotation (free text) and an attribute (key-value pair) with any element. These could then be read by tools producing and consuming SMM-based models. Finally, the Object Management Group have provided an EMF implementation of the metamodel. They propose that users query SMM model instances using OCL or XQuery.

The SMM has received less attention from the research community than KDM [59], although it has been used for several purposes. Berre et al. [60] used SMM to define a number of business process modelling measures, such as value margin. Dahab et al. [61] use SMM to define a 'Computational Energy Cost metric'.

As part of the REMICS EU research project the Metrino tool was created [62]. This provides functionality to: (1) analyse SMM models, (2) add annotations/comments to them, (3) generate reports and graphs to visualise SMM models, and (4) execute metrics defined in a SMM model on other models [63]. The ability to execute metrics is beyond the original design of the metamodel [2], although this is likely to be a valuable feature in many Architecture-Driven Modernisation projects.

2.2.3 Cloud Cost Modelling

Public cloud pricing is complex and difficult to compare [64], although it is typically a major consideration in migrations [18]. Prices vary by regions and consumption, discounts may also be provided for purchasing in advance. A substantial body of work covers: cloud modelling and DSLs (which include pricing), price comparison, and cost reduction.

Leymann et al. [34] developed a model-based method and toolchain to migrate an application to the cloud. As part of this, a cloud metamodel was created to enable

the modelling of a target application and a target cloud. This included some simple cost data, specifically: a compute unit cost, inner data transfer cost, and outer data transfer cost. The cost data was used to determine the optimal distribution of an application's components in a multi-cloud deployment. An implementation of the metamodel (separate from the toolchain) is not provided which limits reuse.

Maximilien et al. [65] proposed a cloud provider and software system agnostic middleware for managing cloud deployments. They created a metamodel to support the modelling of different cloud providers. The metamodel includes several aspects: cloud services (e.g., VMs), available OS images, fees charged, and security features. This metamodel is more flexible for cost modelling than the one proposed by Leymann et al. [34] as it can model a variety of cloud services and each one has an associated fee.

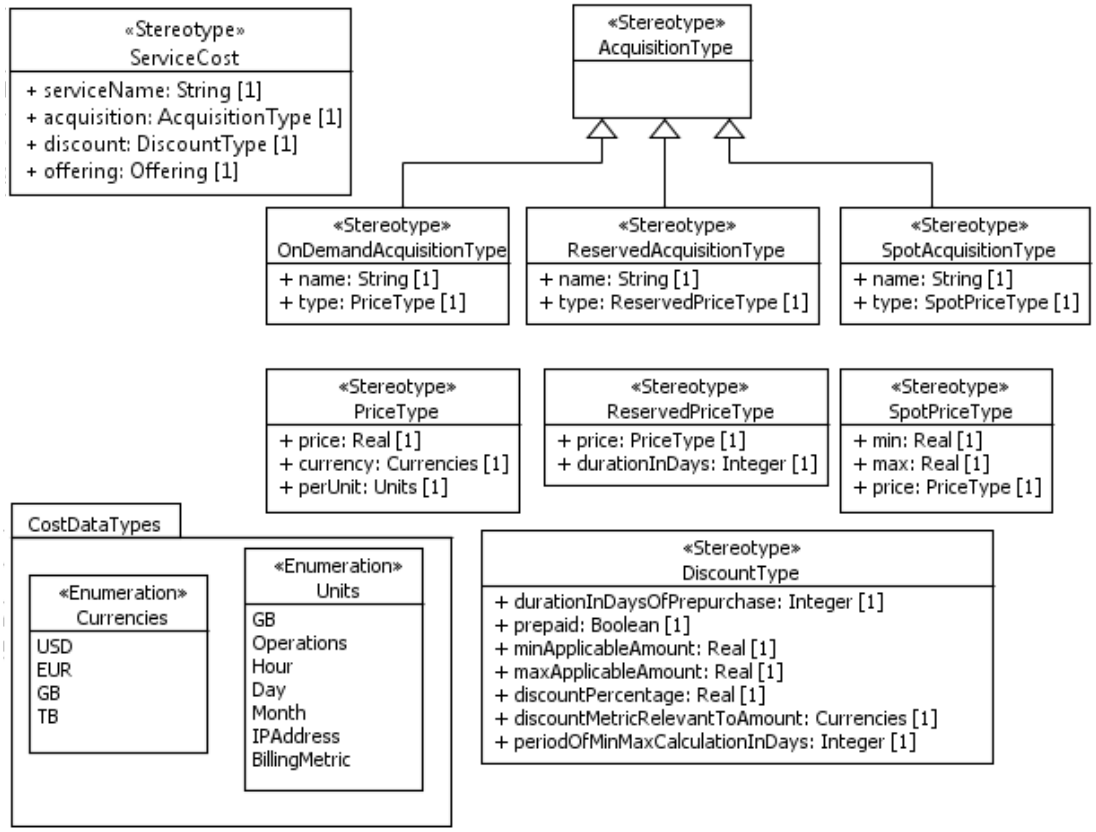

Figure 2.3: The pricing/cost profile of CloudML@ARTIST (adapted from [3])

The CloudML metamodel [26] was developed as part of the ARTIST EU research project — which investigated the migration of legacy software to the cloud. CloudML

is larger than the previous two metamodels and aimed to support IaaS, PaaS, and SaaS platforms. Cloud services, their performance, availability, security, and pricing can be modelled. However, CloudML's goal is to model an entire cloud platform and all its features rather than just pricing. An implementation of the metamodel in UML has been provided, however no complete cloud models (e.g., for Amazon Web Services) have been published.

The pricing component of the metamodel is shown in Figure 2.3. It supports three purchasing options (On-demand, Reserved, and Spot) and discounting. Discounting is an important aspect of cloud pricing models; AWS, Microsoft, Azure, and Google Cloud all offer reduced rates for heavy users and new users. Furthermore, the metamodel supports a range of cloud services, including: compute/VMs, data storage, and static IP addresses. One limitation of the metamodels design is one cloud service will have one price (albeit with variable discounts). In order to support regional pricing variations multiple services would have to be defined. Furthermore, one cloud service can be a composite of other cloud services. For example, an AWS EC2 instance may have storage and IP addresses attached. This is not supported in the metamodel.

2.2.4 Grammar-to-Model Mapping

The first key step in model-driven software modernisation (e.g., Architecture Driven Modernisation approaches and those reviewed later in Section 2.4) is to obtain models from the system's source code. However, developing a tool to automate this is a significant challenge. Izquierdo et al. [66] argue that developing a tool for a single system is time consuming and expensive, more-so for a generic solution. They have proposed a grammar-to-model mapping approach to improve the process of extracting models.

Typically a software system is written exclusively in one or more General Purpose Languages (GPLs), such as Java or SQL, which are defined by a grammar. Secondly, the model a user wishes to create/extract from the source code is typically defined by a metamodel. In [66] a bridge is created between the grammar and metamodel, hence a user writes a grammar-to-model transformation to perform the extraction. This enables a model to be extracted from any system written in the GPL.

The grammar-to-model transformation is defined using a DSL — called Gra2MoL — which was developed by Izquierdo et al. Gra2MoL's syntax is based on ATL and RubyTL so that it is familiar to MDE developers. The approach has been implemented as a tool and applied to an Oracle forms application (written in PL/SQL).

The authors acknowledge that significant research has looked at bridging Domain-

Specific Languages (DSLs) and metamodels. However, they argue in model-driven software modernisation engineers will be primarily dealing with GPLs and face different challenges. Most notably the size of the language and its complexity.

The approach has been compared against: grammar-based bridging (xText), a dedicated parser, and program transformation. Rather than an empirical evaluation Izquierdo et al. have compared the properties of each approach and where they are best suited. They claim that compared to xText and a dedicated parser, the approach significantly reduces implementation time and improves maintainability. This is a result of their approach reusing existing grammars to define the source code and the features of the Gra2MoL language. They claim that their approach makes additional model-to-model transformations unnecessary as it successfully bridges grammar-ware and model-ware.

The argument that grammar-to-model mapping reduces the implementation effort when creating a model extraction tool is well justified. Furthermore, the public release of their tool is a strength of this work. The absence of quantitative evaluation means users cannot understand the extent the approach improves upon existing work. Izquierdo et al. have used grammar-to-model mapping in their later work to model Delphi source code [38] and embedded SQL statements [4], this still excludes a quantitative evaluation. The work has only been applied at a small-scale to limited types of software system.

2.3 Cloud Computing

This section defines cloud computing and its core concepts (Sub-section 2.3.1), reviews the wider field of cloud migration (Sub-section 2.3.2), and the specific problem of resource allocation for migrated systems (Sub-section 2.3.3). Research problems RP1 and RP2 have been identified from (and motivated by) this literature. The majority of the state-of-the-art cloud migration approaches are model-based, therefore building on the work in Section 2.2.

2.3.1 Concepts

Cloud computing refers to computing services consumed on-demand over a network (i.e., as a utility) [12]. These services include: software applications, databases, and virtual machines. The latter allows organisations to deploy their Internet services without large capital outlays in hardware or in-house engineers to manage it. Over and

under provisioning of infrastructure can be avoided. Using cloud services a compute job which might take one month on a single quad-core server, could be performed in a day on a cluster of 30 servers for the same cost.

Cloud services are typically categorised as: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS) [67]. *IaaS* includes hardware resources (e.g., for computing, storage, or networking) which are typically encapsulated/abstracted using virtualisation. A *PaaS* is a high-level environment to build, test, and deploy custom applications. Users do not need to consider hardware provisioning or scaling as this is handled by the environment, although restrictions typically exist on how the software is designed and written. Finally, *SaaS* refers to software delivered over the Internet. Services in all categories employ a usage-base pricing model.

These categories are often described as levels, where IaaS has the least abstraction from hardware and SaaS has the greatest [67]. Typically a PaaS will be built on an IaaS, and a SaaS will be built on a PaaS or IaaS. Opinion varies on whether this inheritance is a mandatory part of each level's definition e.g., [68] and [12]. The boundaries between each level are also difficult to establish, as a result other taxonomies have been proposed [69].

A cloud service (in any level/category) will have one of the following deployment models: public, community, hybrid, and private [68]. A private cloud is used exclusively by users within a single organisation, although it may be owned and operated by a third-party. They are typically used by larger organisations with multiple business units, and can mitigate some security concerns [70]. A community cloud is a private cloud which is shared by multiple organisations, while a public cloud can be used by anyone. A combination of private, public, or community clouds is referred to as a hybrid cloud. A combination of two or more clouds with the same deployment model used together is a federation [71]. Hybrid clouds and federations are especially valuable when one cloud cannot meet peak demand i.e., for cloud bursting.

2.3.2 Migration

Challenges

Dillon et al. [72] have identified six core challenges to be considered when migrating systems from an in-house to cloud environment. The first is security, as outsourcing infrastructure has inherent risks such as data being leaked by a provider or data being compromised during transfer [73]. In the cloud there is the additional concern of multi-tenancy, where an attacker can exploit instance placement to become co-resident with

a victim, allowing them to perform side channel attacks [74]. However, this type of attack has yet to be used on real-world systems. A potential concern for community clouds is the shared reputation between platform users, if a botnet was to be created on the architecture or it was used for other illegal purposes by one user, then another organisations system may be affected by IP blocks by networks [72]. A final challenge is ensuring the complete deletion of data, due to the automated backup and replication methods employed by cloud providers data may still exist after it has been deleted [75].

The next challenge identified by Dillon et al. [72] is the costing model. While running a system on the cloud will reduce infrastructure costs [76], it is likely to increase data transmission costs therefore the cloud may not be economical for some types of system. A key issue is also determining what the cost will be of an application once it is deployed in the cloud, as the number of virtual machines / instances required may not be known. The addition of security measures (like multi-clouds [77]) or modification of software and deployment scripts to use cloud APIs, is another cost area. It is therefore important to carefully analyse the costs of a migration.

Identifying a charging model is another challenge to address, both for cloud providers and consumers [72]. The diverse nature of services and instance types offered by cloud providers results in a more complex charging model for providers. Organisations transitioning from selling a one-time licensed software product to a subscription-based software service will have to determine what they now need to charge their users.

Service Level Agreements (SLAs) will need to be considered [72]. In many situations organisations must ensure the performance of their systems running in the cloud, even though they do not have control of the underlying architecture. It is therefore challenging to guarantee a SLA.

The majority of cloud providers have different APIs for accessing and controlling their services, furthermore the services offered may also be different [72]. The heterogeneity of cloud services results in a lack of interoperability and the issue of vendor lock-in. Tactics for solving this issue include the use of an intermediary layer (e.g. jclouds [78]) or standardisation (e.g. OpenStack [79]).

A final challenge for organisations considering cloud adoption, is deciding what to migrate to the cloud (e.g. storage, compute resources, etc.) [72]. The decision is not straightforward and will require analysis of the costs, benefits, and risks. It should also be noted that non-technical challenges also exist, such as the organisational issues described by Low et al. [80], the socio-technical issues described by Khajeh-Hosseini et al. [19], and the regulatory and legal issues described by Marston et al. [81].

In the following sections of this literature review, cloud migration methodologies

are analysed to determine their success in addressing these challenges. The aims and objectives of methodologies will also be noted to identify the most significant and common challenges.

REMICS

REMICS was an EU FP7 project which finished in 2013, and produced a model-driven re-engineering and cloud migration methodology with an accompanying set of tools [82]. Mohagheghi et al. [25] define the project's objective as the production of model-driven tools and methods to support organisations in modernizing and migrating their systems to the service cloud paradigm. Where the service cloud paradigm is the use of cloud computing and service-orientated architecture to develop software-as-a-service systems.

The role of the methodology is to provide a set of recommended practices to guide organisations in their migration projects, these are grouped into activity areas [83]. REMICS has seven phases: requirements and feasibility, recover, migrate, validate, supervise, withdrawal, and interoperability. Starting with the Requirements and Feasibility activities, the requirements of the re-engineering project are established, the feasibility is assessed, and planning is performed. Next is the recovery activity where the architecture of the system is extracted, during this phase models are produced for business processes, business rules, components, implementation and test specifications.

During the migration, the new service orientated system is built using the models as a starting point. The validation activities include the functional validation of the system as well as validation of performance, reliability, and security. The supervise activities focus on monitoring the performance of the system and managing the resources it is running on. The interoperability activity area conceptually spans the requirements activity area through to the supervise activity area, and is intended to manage and mitigate the issue of vendor lock-in or API changes impacting the system.

The tools which accompany REMICS provide support for: requirements management, knowledge recovery, migration to service-orientated architecture, model based testing, and system evolution. The tools are a mixture of open source and proprietary applications, which are external to the REMICS project with the exception of CloudML [84].

As a consequence of the large tool set and the effort of the development community in extending the open source tools, there is support for many programming and modelling languages. Modelio provides the reverse engineering of Java applications to

UML models, as well as code generation of Java. BluAge can reverse engineer Cobol, Delphi, C#, VisualBasic, and others to UML in addition to Java. Furthermore some of the research outputs of the REMICS project have been incorporated into these tools.

Examination of REMICS uncovers areas for potential future work, as well as the opportunity for different approaches to be used. The authors state that "Knowledge discovery is often limited to reverse engineering of legacy code" [82]. However, the REMICS tools and practices do not consider incorporation of any existing artefacts into activities relating to knowledge discovery. It has been seen in Science Warehouse and other external case studies [85], that there is an abundance of existing knowledge and artefacts within a company. For example, this could include UML models, textual design documentation, developer knowledge, monitoring data, or log files. In situations where these types of artefacts are available then the quantity of information which needs to be discovered through the reverse engineering of code is less. Here, the challenge is to combine existing knowledge and reverse engineered knowledge of a system.

The scope of REMICS is large, and as mentioned previously the REMICS tool-set includes model based testing tools, specifically Fokus!MBT [86] and the RedSet [87] tool which are used within the validation activity area. Whilst model based testing is valuable, it can be argued that including it in a cloud re-engineering and modernisation methodology is unnecessary as it increases its complexity. A barrier to adoption of model based approaches is often the perceived complexity and learning curve required in using them. Furthermore the model-based testing tools and approaches in REMICS are often not specific to cloud re-engineering and migration.

Ultimately a balance should be struck by methodologies like REMICS in regard to scope. Large methodologies which combine many tools can be considered inherently more complicated due to their size and number of possible routes from a legacy system to a cloud-based system. In contrast smaller methodologies may be very specific and therefore not be applicable to a large majority of systems.

ARTIST

ARTIST is an EU FP7 project which ran between 2012 and 2015 [88]. The project was coordinated by ATOS Spain, in conjunction with Vienna University of Technology, Institute of Communications and Computer Systems (Greece), French Institute for Research in Computer Science and Automation, Engineering Ingegneria Informatica, Fraunhofer IAO , Spikes NV, SparxSystems Software, and Tecnalia.

The project proposes a model-driven modernisation (MDM) approach for migrat-

ing legacy software to cloud-based software, while the intended outcome is a comprehensive tool suite [55]. MDM is defined by Bergmayr et al. [55] as the process of semi-automatically discovering models which represent the legacy software, then transforming the models until the software satisfies the modernisation requirements. At the core of ARTIST is the idea of pre/post migration phases, where goals and requirements are defined before migration and are then verified afterwards.

The methodology states that the legacy software must be analysed to establish well defined migration goals during the pre-migration phases. Next is the migration phase, where models should be reverse engineered from the legacy software, these models include all of the specifics imposed by the platform and are referred to as the legacy platform specific model (PSM). The legacy PSM should then be transformed into a higher level representation called the platform independent model (PIM) to abstract from such specifics like the software runtime environment. The PIM is then transformed to suit a particular cloud paradigm (e.g. IaaS or SaaS) and to achieve the migration goals, this results in the 'cloudified PSM'. The final step in the migration phase is to transform the cloudified PSM into the migrated software. In the post migration phase equivalence tests are derived from the PSM to verify the migrated software. Furthermore the software is analysed to ensure that the goals identified during the pre-migration phase have been met. It is proposed that the verification of a migrated systems is best performed by semi-automatically generating test cases instead of any formal verification. Bergmayr et al. [55] state that formal verification is impractical on large systems.

Bergmayr et al. [55] argue in favour of the assumptions which underpin ARTIST. In particular [55] states that a model driven approach was selected to automate the various steps in the legacy-to-cloud migration process. It is claimed that the use of MDM lowers the barriers of cloud adoption by reducing the effort required for migration. However, research has shown that the MDE field has itself many barriers to adoption in industry [89] and these do not seem to have been considered at this stage of the project.

At the inauguration of the project, some key challenges were identified which would be addressed. One challenge was deciding whether modernising legacy software is preferable to replacing it, and whether a cloud platform could satisfy the expectations of the organisation [55] [90]. However, the decision on whether modernisation is the best strategy is perhaps best taken on a component level, as is advocated in REMICS [83]. It is unlikely that legacy software has no components which can be re-used, wrapped, or extracted. Re-using as much of the legacy software as possible should be a key priority of any methodology.

Another research challenge to be addressed in the project is defined as the 'Operationalization of the software modernization' or in other words performing the migration in practice. It can be argued that it is critical to keep this in mind when developing such approaches, as this will increase industry adoption.

There are some areas of the currently published ARTIST methodology [55] which do not clearly identify their cloud re-engineering and migration specific aspects. An example of this is the storage of migration artefacts: "Modernization artefacts are stored in a central repository to foster reusability across all modernization tasks". In many software development projects, processes to share artefacts are likely to be in place, such as version control systems and wikis.

The ARTIST methodology and tool-set are verified using the industrial partners in the project, and through case studies on on-going commercial projects. This was also a strength of the REMICS project (discussed previously in Section 2.3.2). Reviewing ARTIST and REMICS shows that neither provide explicit support for database or data migration challenges. However, both do consider the feasibility of migrating the software components of the system. This reinforces the 'costing' challenge identified by Dillon [72] (discussed in Section 2.3.2), which argued that determining the financial feasibility of a cloud migration is a major challenge. A gap in both methodologies is identifying cloud database migration and future running costs (i.e., research problems RP1 and RP2).

2.3.3 Resource Allocation

CloudMIG

CloudMIG is a model-based approach for semi-automating the migration of enterprise systems to scalable and resource efficient PaaS and IaaS deployments [21]. The authors aim to address the heterogeneity of cloud environments by providing a generic migration approach. They explain that previous work in this area has multiple shortcomings, including: a restriction to single provider, a lack of automation, no consideration for the efficient use of cloud resources, and no support for automatic scaling.

A key premise for CloudMIG is that running existing software in the cloud involves extensive re-engineering [21]. Without this, the scalability issues an organisation was having in non-cloud infrastructure, will remain. The six main activities in the CloudMIG approach are:

1. Extraction of a KDM model which defines the architecture and an SMM model for relevant metrics. The latter focuses on software layer metrics, including: service

invocation rates over time, request data size, and memory footprint.

2. Selection of a cloud provider, which is defined according to a Cloud Environment Metamodel [91].

3. Generation of the target architecture.

4. Manual adaptation of the target architecture to accommodate user-specific requirements.

5. Evaluation of the target architecture using CloudSIM.

6. Transformation of the legacy system to match the target architecture.

These activities are supported by the CloudMIG Xpress tool [92]. While this tool is publicly available, its source code has not been published. This prevents re-use and extension. Another shortcoming of CloudMIG is its lack of consideration for cloud database migration. The authors identify horizontally scaling the data persistence layer as a challenge in [21], reaffirming research problems RP1 and RP2 which are addressed in this thesis. Furthermore, the KDM and SMM model extraction functionality in CloudMIG Xpress cannot model the database.

Log2cloud

Log2cloud [93] is a technique for predicting the cost of virtual machines that a cloud deployed application requires to achieve a given SLA. The technique is based upon queueing network theory and analyses the logs of existing request handling applications, such as email and web servers. Log2cloud has been implemented as an open source Java library to allow easy integration with existing tools.

A cost prediction is produced through a two stage approach, the first stage is the generation of a formal compact profile of the resources required by the application, called the 'probabilistic resource usage pattern'. The goal is to abstract out the irrelevant raw data. The second stage is to analyse the resource usage pattern using the open source 'probabilistic pattern modelling tool'.

As inputs Log2cloud requires the SLA, the virtual machine performance, and the application log. This provides the total number of VM costing time intervals (i.e. instance hours) that the organisation needs to pay for. The approach was evaluated using the University of Zaragoza's webmail system, which was well-suited as it logs all the requests which are received.

The authors argue that a significant limitation of existing cost calculators, such as those from providers, is that they require users to know the resources they need. Log2cloud overcomes this for software components being deployed to cloud virtual machines. In this thesis, research problems RP1 and RP2 address this same issue. The approaches presented in Chapters 3, 4, and 5 overcome this for the database layer. Many unique challenges exist at this layer, as discussed later in Section 2.4.2.

CloudML

CloudML is a Domain-Specific language (DSL) for cloud provisioning which is capable of expressing the resources and services that are either required for a system or available from a cloud provider [94]. The aim of this work is to address the heterogeneity of clouds and the need to have provider-specific scripts to deploy a system [95]. This helps to address vendor lock-in through an additional layer of abstraction.

CloudML was developed as part of REMICS through the funding of a masters thesis [94], and includes a metamodel to define the language and an engine which performs the provision according to a CloudML model. The engine connects to the provider using its 'cloud-connector' component which supports several interface libraries, although the default is Apache jclouds [78].

Brandtzaeg et al. [95] identify the following challenges for the platform-agnostic automated deployment of systems:

- Heterogeneous interfaces;
- Platform specific configuration;
- End-user reproducibility;
- Sharing provisioning scripts;
- Robustness (provisioning can fail due to the cloud provider);
- Run-time dependencies (IP addresses allocated on provision are needed for the configuration of the system).

The key contribution of CloudML is the addressing of these challenges, and thereby improving interoperability. CloudML is XML-based and within the language three groups of schema exist, these are for: resource, service, and requirements description [96]. Resources are described in terms of CPU, RAM, storage provision, geographical location, architectural type, and network performance. Services are described similarly

to resources with the addition of information on the type of service provided. The requirements then define what resources/services are needed for a system, as well as it's location.

The success of the language and engine in addressing these challenges was examined empirically [94]. A three node topology was used with two front-end nodes (with a dual-core CPU and 1GB of RAM), and one back-end node with 500GB of storage. The CloudML code for this configuration was given to the CloudML engine. The results showed that the engine successfully launched two Amazon c1.medium instances and a t1.micro instance with a 500GB EBS volume attached. This was the cheapest configuration that satisfied the requirements at that time.

Limitations and possibilities for future work were identified through the review of CloudML. The approach is potentially of high value, as there is minimal work in the area of model-based cloud provisioning. Furthermore the extensible design and open-source nature of the engine makes it easy for organisations to develop the approach. Although one potential limitation is the lack of consideration for performance. Defining compute requirements in terms of RAM, CPU performance, and architecture is limiting. Research has shown that the underlying CPU model for a virtual machine can strongly affect the performance, and instances of supposedly equivalent types can have different CPU models and therefore significantly different performance [97]. Large systems which are being migrated between cloud providers may see different performance on different platforms. Although in practice an organisation is likely to detect this during testing and adjust the number of instances to ensure that SLAs or other performance requirements are not breached.

An additional issue which is not fully described in the literature is the extraction of the PSM defining the services and resources that a cloud provider offers. It is expected that cloud providers will describe their services/resources in CloudML and provide this to their customers [96]. It is likely — although not stated — that for the purposes of the verification that CloudML descriptions of the providers were created manually. This is one overhead if the approach was to be adopted by an organisation, particularly considering the frequency with which providers revise their offerings.

CloudML is of relevance to organisations considering migration (such as Science Warehouse) to the cloud or the use of multiple clouds and are concerned about the issue of vendor lock-in. The engine connects to a variety of cloud platforms using its 'cloud connector' module, this allows CloudML to connect to several providers through a common interface [94] and acts a bridge between the providers and the engine. The module follows a faade pattern [98] which can wrap any external library(ies).

2.3.4 Case Studies

In this subsection cloud migration case studies and general re-engineering case studies which are of particular interest are discussed. The identification of process and logistical challenges in cloud migration is important to ensure the practicality and relevance of the research in this thesis.

Chauhan and Babar [99] describe the migration of Hackystat, an open source framework for the collection and analysis of software development process. This required the modification of the delivery method from a software product to SaaS. The purpose of this migration was to support large distributed software development teams.

The migration of Hackystat started with the identification of migration requirements and objectives. The migration was then performed in four steps, first the framework was evaluated for scalability, next it was evaluated for orchestration, then components were identified for re-factoring, and finally the solution was evaluated against the target cloud environment. All of these activities are performed manually. The scalability evaluation involves the identification of any stateless components, while the orchestration evaluation assess whether the framework can load-balance itself when deployed on several nodes. If any issues are identified in either of these steps then they are re-factored. The final evaluation step involves deploying the application on a target cloud platform, such as EC2, and ensuring there are no issues.

The case study in [99] focuses on the technical decisions which need to be made, and steps which need to be taken. Chauhan and Babar [99] conclude that: a software system which consists of stateless components is easy to migrate, a SaaS system should not be designed using technologies whose components cannot be hosted on separate clouds, and finally that the business objectives should be carefully analysed to produce requirements prior to migration.

In [100] the experiences of the ARNO project are described, in which an e-commerce application is successfully migrated from a mainframe architecture to UNIX servers. This migration is of interest as it is one of the few recent papers which advocates the use of one-off bespoke tools to perform the migration. Furthermore, a number of interesting conclusions are made.

The application being migrated in the ARNO project was a large-scale system used by Travel agents in Germany for making a variety of travel bookings and reservations. The reason for the migration was the replacement of the programming language the system was written in as it was out-dated and therefore difficult to find developers with these skills. It was argued by Teppe [100] that a key factor in the success of the

project was the use of tools which were created solely for the migration, as this allowed a higher level of automation. Teppe also stated that the development of the tools was outsourced, so it is difficult to conclude whether the tools used were extended versions of generic tools. When it came to the migration of the data, from the 'high performance file handling system' used by the mainframe to an Oracle database, middleware was implemented to translate queries and error codes. The middleware approach proved effective as the original file system had a limited API and simple queries, therefore translation was straightforward.

While extensive technical details were provided, the rationale for creating one-off tools was minimal making it difficult to determine the advantages and disadvantages. A interesting point made by Teppe is that the longer a migration lasts, the more the migrating system will change, therefore the goal should be to perform a migration project as quickly as possible.

The cloud migration case studies in the literature are presented in a variety of different ways, and have a variety of focuses. Case studies have been identified which focus on moving big data into the cloud for analytical purposes [101], and which focus on the socio-technical aspects of enterprise system migration [19]. One practice which was not seen is the migration of an unchanged system followed by the incremental re-engineering of the system to exploit the benefits of the cloud. This may be due to small-scale or routine migrations not being reported.

2.4 System Modernisation

This section investigates work on system modernisation which is not specific to cloud computing. It includes the software layer (Subsection 2.4.1) and database layer (Subsection 2.4.2). The research problems being addressed in this thesis focus exclusively on the database, although organisations will typically be migrating the system as a whole. It is therefore important to identify standardised metamodels in this related area, and any approaches which might be used alongside those from this thesis.

As discussed previously, migrating a system to the cloud typically requires some re-engineering to fully exploit its properties. Subsection 2.4.1 starts by reviewing some leading model-based refactoring approaches. Next, it presents the leading MoDisco model extraction tool which can produce models from software source code. These all use the Knowledge Discovery Metamodel.

Subsection 2.4.1 starts by defining the key database terminology and concepts necessary to understand the work in this field. This includes the state-of-the-art ExSchema

and Schemol database modelling approaches. The purpose of this section is to establish the limitations of these existing approaches, which forms the basis for research problem RP3. It also establishes whether they consider the cloud-relevant properties of a database (research problem RP4).

2.4.1 Software Layer

Model-Based Refactoring

Refactoring is the process of improving internal quality while maintaining behaviour [102]. It is a key activity when modernising legacy systems as it improves maintainability and facilitates code re-use [103]. Durelli et al. [102] claim that despite this, leading model-based modernisation approaches like ADM do not support refactoring. As a result they developed a refactoring catalogue for software system models conforming to the Knowledge Discovery Metamodel.

The catalogue contains seventeen refactorings, such as: class extraction, encapsulation, and renaming. These are a subset of those published by Fowler et al. [104]. Each refactoring has been adapted or specialised for use on KDM-based models. The catalogue has been implemented as an Eclipse plug-in called the *Modernization Integrated Environment*. As the refactorings are applied on a model — previously extracted from a system's source code — they are programming language independent.

The contribution of this work is the addition of refactoring to the ADM workflow (as discussed in Section 2.2.2). Furthermore, the authors investigate if refactoring is effective in improving internal quality of a source code model. This is of interest because refactorings are often developed for a specific programming language or a type of programming language.

Durelli et al. only applied three of their refactorings from the catalogue against eight KDM-based models. The models were extracted from open-source Java applications. They found their refactorings improve the quality of the system, as measured using the CAMC [105] and SCC [106] metrics.

This work only considered KDM-based models extracted from Java applications. It could have been strengthened by looking at models from other object-orientated languages, and languages from other paradigms. The proposed catalogue and tool is still likely to be of interest for organisations and researchers investigating model-based modernisation. It is the first — and only at the time of writing – published work which investigated refactoring in model-based modernisation.

SMART

SMART (the Software Migration and Reuse Technique) is a process for establishing the feasibility and performing planning for legacy-to-SOA (Service Orientated Architecture) migration. It was developed by the Software Engineering Institute at Carnegie Mellon University [107]. Migration of any kind requires a detailed analysis of the feasibility, risk, and cost; which is the challenge SMART is targeted at, with an emphasis on SOA. Many of the activities in the process are likely to be applicable to a wider range of migration projects. Common with approaches in the field of re-engineering and migration, SMART is evaluated through a case study where it has been used on a real-world migration project.

In SMART the emphasis is on wrapping components extracted from legacy systems with the least amount of re-engineering. Information is gathered on legacy components, target environments, and potential services. This produces a preliminary analysis of the viability of the migration, an analysis of the migration strategies available, and estimates of the costs and risks involved.

SMART consists of six activity areas: establish context, define candidate services, describe existing capability, describe target SOA environment, analyse the gap, and develop strategy. Figure 2.4 shows the flow between these. In the 'Establish Context' activity a number of important project aspects are identified, such as the stakeholders, who is driving and paying for the project, mission goals and tasks. In the 'define candidate services' activity, all the possible services are identified through developer knowledge and hands-on analysis of the system. The 'describe existing capability' activity looks at defining the system in terms of age, platform, existing issues, and other similar aspects. Describing the target environment includes the platforms, API libraries and technologies, and other standards which may be used. In the 'Analyse the gap' the effort, cost, and risk for producing the candidate services are identified. Finally a strategy is developed for the migration where the issues are defined, planning is performed and any training requirements within the organisation are identified.

After the 'establish context' activity is the critical decision point to proceed or cancel the migration project. A number of characteristics of good and bad legacy system candidates for SOA migration are given. Good characteristics include: the goals are clear and can/have been identified, candidate services and potential consumers have been identified, there is functionality in the legacy system which is stateless in nature, and the stakeholders are available to take part in the project.

The activities in SMART make use of the Service Migration Interview Guide (SMIG),

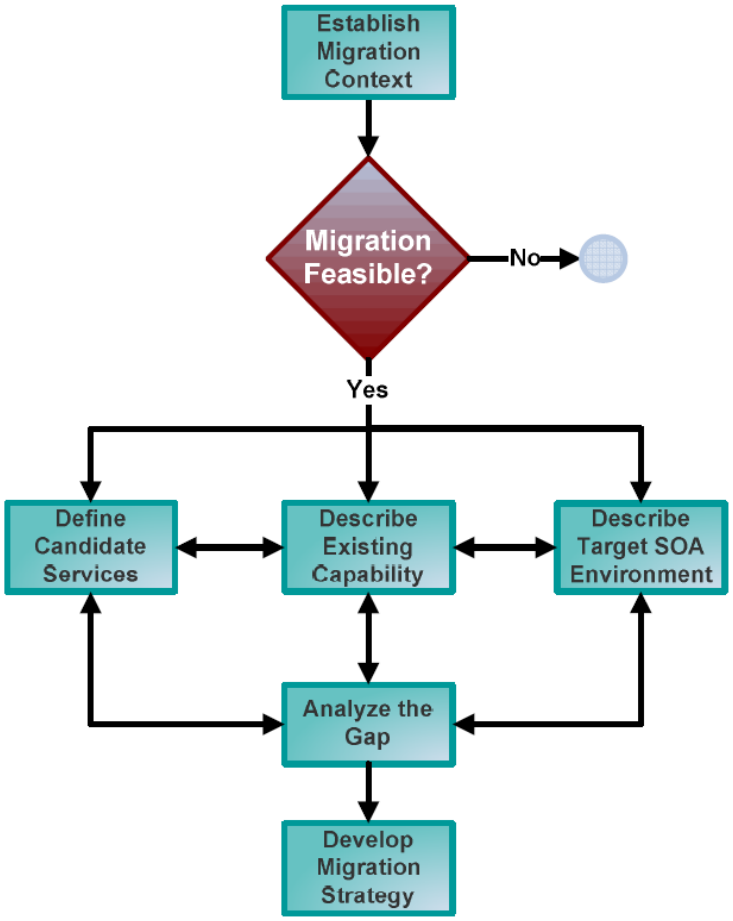

Figure 2.4: The SMART activity areas

which is described as an instrument to focus the discussion with stakeholders. The SMIG along with the high level of detail in the description of SMART, makes it a particularly practical approach. The validation of the process was performed using a small system (13,000 lines of code) with a 'moderate' level of documentation. One area of future work would be to establish how well the approach scales and what changes would be necessary to support legacy systems with code quality issues or an eroded design.

MoDisco

MoDisco (Model Discovery) is an Eclipse modelling project for reverse engineering, which was developed under the MODELPLEX EU project on model driven solutions

for complex systems [49]. The aim of MoDisco was to provide a generic and extensible reverse engineering framework, this was achieved through the use of OMG standards and extensive technical documentation [53]. Bruneliere et al. [53] describe Model Driven Reverse Engineering (MDRE) as the transition of the heterogeneous world of implementation technologies to models, however the key difficulty is this heterogeneity which makes the use of standards to achieve extensibility even more important. The OMG standards used in MoDisco are KDM, SMM, and ASTM.

Ulrich and Newcomb [49] describe the architecture of MoDisco as a three layered design, with a use case layer, a technology layer, and an infrastructure layer. At the use case layer are the functions that a user might perform (e.g. Quality assurance to check the code quality and other properties, migration to a different language or architecture, and re-factoring). The technology layer provides support for the technologies (e.g. Java or SQL). The infrastructure layer consists of knowledge components (KDM, SMM, and ASTM), and technical components (abstract discoverers and model browsers) to support the upper layers. To extract a model with MoDisco, the first step is to create or obtain a metamodel. Next, a discoverer must be created which builds the model according to the metamodel. Furthermore, generators can be created for code generation from models.

MoDisco can be easily installed through Eclipse and supports Java, JEE, and XML [108]. Metamodels, discoverers, and generators are provided for each. Java can be transformed to a KDM or UML model. Furthermore, the extensible nature of the framework means support for other technologies can be added.

The approach used by MoDisco has the advantage of being based upon standards, and implemented within Eclipse. Both of which are likely to increase industry adoption. While, Eclipse integration makes MoDisco more familiar and accessible to the large population of Java developers who use the Eclipse IDE. However, the disadvantage with a plug-in is that there are more extension restrictions than with a stand-alone tool.

2.4.2 Database Layer

NoSQL and Polyglot Persistence

The NoSQL movement has seen new types of database come to prominence, driven by the need to store large amounts of data on clusters of servers, something which relational databases were not designed to do [109]. NoSQL databases can be categorised as key-value stores, column-oriented databases, and document databases [110].

Graph databases are also considered a type of NoSQL database, however unlike other NoSQL categories they were not developed to overcome scaling complexities in relational databases [111]. Sadalage and Fowler [109] introduce the idea of polyglot persistence and describe how different types of database are suited to storing particular data. They argue that developers and designers should no longer turn solely to a relational database, but instead should identify the type of data they are storing, the level of consistency required when dealing with multiple database nodes, and the types of queries that are expected. Many legacy systems are built around a single relational database which is used for storing all the types of data in the system. However given the scalability challenges that using a relational database brings and the sophisticated techniques which have to be employed [112], using the database which best fits each type of data is attractive.

When using several databases within a system the benefits that they bring should be considered. First of all, relational databases are suited for systems which require strict consistency (ACID compliance) or need to store structured data [113]. Graph databases are used for storing relationships between entities [111]. A Key-Value store is suited for simple data models with few interdependencies and constraints, where speed and scalability are the most important factors [114]. Some typical uses include: storing session information, user profiles/preferences and storing shopping cart data [109]. While a document store is suited towards big data storage and good query performance [110]. Finally a column-family store is best suited for storing large amounts of structured data in a flexible way, but without strict consistency or transactions [115]. Use cases for column-family stores include event logging, content management systems, and blogging platforms [109]. Ensuring a database is used for the scenario to which it is best suited simplifies queries, data models, and data access code. The disadvantage with most key-value, document, and column-based data stores is their eventual consistency property, where if a write is made there is no guarantee that the next read will always reflect this [116].

A system with relational, graph, key-value, document, and column-based stores will have a more complicated data access layer compared to one with just a relational database. Furthermore, deploying a system to staging and production environments is more challenging when dealing with a mixture of databases.

Analysing the data a system is dealing with will normally highlight differences in requirements. For example, data from transactions in an e-commerce application is likely to be best suited to an ACID-compliant relational database. However, the items that a user has in their shopping basket could easily be persisted in a key-value store.

The division of data into different databases can promote reuse and facilitate a service orientated design. NoSQL databases can be wrapped within services, which can provide the sophisticated security features that are standard on enterprise relational databases. These services can then provide data to new systems in a straightforward way.

Database Reverse Engineering

The lesser-considered artefact of software systems is data (and the database containing it), which can also suffer from erosion. Pérez-Castillo et al. [117] propose an approach for the extraction of minimal database schemas through static analysis of a system's source code. A 'minimal' database schema is one where duplicated or unused tables and columns which exist in the database have been removed.

The approach creates a 'SQL sentences' model by using a parser to discover the SQL statements which are embedded in the code. The discovered statements are then combined to create a 'SQL sentences model', which is transformed into a KDM-based model. A series of rules — given in the paper [117] — define how this model is produced from the SQL. The use of KDM abstracts any platform specifics such as the type of relational database used. The approach is verified using a case study on an internal system at the author's university.

This approach has some possible limitations which are not discussed by Pérez-Castillo et al., such as the use of Object Relational Mapping (ORM) frameworks, the use of prepared statements, and the effectiveness in other types of legacy system. It is implied throughout the paper that the SQL statements are strings within the source code, which is unlikely in production systems due to the SQL injection vulnerabilities this can introduce. The intranet which was used for verification contained 72,000 lines of Java and had an Oracle relational database with 140 tables. It was reported that the schema was reduced to around 80% of its original size. It is not noted whether every table used by the system was identified from the static code analysis. This would be an interesting aspect to investigate as large legacy systems will have been written by many different people, in different ways, over a long period of time, and it is unlikely that 100% of the tables could be identified in all systems.

In Java there is the possibility to query a database via JDBC using hard-coded SQL strings, Prepared Statements, ORM libraries such as Hibernate, StringBuilder to construct the SQL string, and SQL generation libraries such as jOOQ [118]. This diversity is a key challenge, however the approach developed by Pérez-Castillo et al.

has the potential to remove a significant amount of the unused tables and columns which are common in legacy databases without a large manual effort. In a legacy-to-cloud migration scenario a reduced schema will typically result in reduced data transfer costs.

Data Migration

This subsection reviews several approaches and challenges in the area of data migration. One such approach is Slacker [29], an end-to-end database migration system which uses COTS backup tools to perform live database migration between servers. The Slacker prototype focuses on MySQL, and is written from the perspective that cloud providers would be the primary users. However, it is expected that the technique used by Slacker to migrate the data will have uses outside this context.

Slacker can perform two types of migration, the first of which follows a stop-and-copy technique [29]. Here the database is either shut down or a global read lock is applied. The data is then exported using a tool such as mysql-dump to a file, which can be imported into the target database. This approach is slow due to the overhead of reimporting the data, and the downtime required is a significant problem.

The second type is live migration, which is the primary contribution of Slacker. This is performed using the open-source hot backup utility Percona XtraBackup [119]. Although XtraBackup is specific to MySQL alternatives exist for most databases, such as RMAN [120] for Oracle. The migration process is made up of three steps: the first is 'snapshot transferring' where the snapshot data generated by XtraBackup is streamed to the target server on the fly, the second is 'delta updating', and the third is 'freeze-and-handover'. During the delta updating step Slacker applies several 'rounds' of deltas to gradually bring the target up-to-date with the source. This is necessary as the snapshot will be out of date due to the time spent preparing it. Once the delta size has reduced to a sufficiently small amount the freeze-and-handover takes place, here the source database is frozen and the final delta is applied. Barker et al. [29] reported that this took less than a second during their testing.

Slacker is named after the 'migration slack', which is a core concept of the live migration. The slack is the resources that can be allocated to the migration task without violating any service level agreements or performance requirements a system may have. As the load on a database will vary depending on system usage, the slack available is not constant and therefore the migration should be dynamically throttled. The throttling is achieved using the pv utility which can limit the amount of data

passing through a Unix pipe, pv is used on the output stream from XtraBackup. The utility is controlled using a proportional integral derivative controller (PID) which takes the current transaction latency of the database as an input, and provides instructions for pv utility as output. The PID will then dynamically adapt the migration speed so that transaction latency does not exceed a permitted level.

While the use of an existing backup tool reduces the implementation effort, it is also limiting. It can be argued that if the system being migrated has a polyglot persistence model then it would be impractical to coordinate several different tools so that the freeze-and-handover happened concurrently. Furthermore it is only possible to use Slacker if both the source and target database are the same, as including re-engineering (for example to accommodate Oracle to MongoDB migration) somewhere in the approach would present a large challenge. Despite the intention of the approach being used by cloud providers to move databases between nodes, nothing within Slacker would prevent it being used outside the cloud or to migrate databases running on top of compute instances.

The cloud data migration literature has a focus on migration within cloud providers, and there is minimal literature on moving between in-house and cloud [29] [121]. The literature which does focus on in-house–to–cloud is normally targeting scientific workloads and big data analytics [122], perhaps due to the time-critical nature of these problems. The only model-based data migration approach assumed that data was being moved between two versions of the same system, and that these had been developed using MDE [123].

A second interesting data migration approach is proposed by Kotecha et al. [124], who describe a solution for the migration of data from one cloud to another. They identify three data migration tasks: data extraction from the source database, mapping to the target database, and the import of the data into the target database. In a live migration approach like Slacker [29] these tasks would be performed repeatedly to synchronise the database, however Kotecha et al. only consider a 'stop and copy' approach.

During the mapping task the migration of stored procedures and queries needs to be considered in addition to the data. The primary contribution of [124] is a query migration framework, as it is noted that this is the area of migration which has received the smallest amount of attention.

This query migration focuses on SQL-to-GQL (Google Query Language). GQL is used to query the Google Cloud Datastore, part of Google's App Engine product suite [125], and uses a SQL-like syntax. The core difference between these languages

is the lack of update, insert, or delete statements, as well as no support for joins [124]. However, these similarities make SQL-GQL query migration more straightforward than would be the case for other types of NoSQL data store.

In [124] a basic framework is proposed which contains: a syntax analyser module, a mapping generation module, and GQL generation module. The syntax anaylser validates the SQL syntax, while the mapping generation module maps SQL constructs to GQL constructs according to a mapping table. The authors do not identify whether the SQL is extracted from the application being migrated, however it is implied that this is done manually. Despite the limitations of this work, such as the language similarities and the lack of consideration for prepared statements and ORM frameworks, it is still of interest due to the minimal amount of work in this area.

A third paper of interest in the area of data migration is by Schram and Anderson [126], who provide a detailed account of the transition from a relational to a polyglot persistence model within a research application. They state that a significant challenge when transitioning from a relational data model to a polyglot persistence (hybrid) model, is the transformation of an existing systems data model.

The research application [126] was used for harvesting tweets from Twitter when a public emergency or disaster occurred. Unlike many re-engineering projects which deal with legacy applications, this application was modern and had only been deployed for two years. It had been written in Java with Hibernate used to access a MySQL database. MySQL had been chosen because it was mature and familiar to the developers, the aspect of scalability had been considered but at the time of the design a relational database was deemed sufficient. Although the scale of the data which the application needs to store has since grown sharply, causing the developers too look at alternatives. This is likely to be a common situation facing many other systems.

Young greenfield systems typically have a well defined and logical system architecture, as was evident in the description Schram and Anderson [126] provided. The system was separated into layers and services, the layers included: database, persistence, services, and application. The abstract interfaces between the layers allowed components in the persistence layer to be changed from MySQL to a MySQL and Cassandra hybrid without impacting the rest of the application.

The step which required the most consideration was the de-normalisation of the existing data model. In Cassandra there is no way to enforce relationships, and data should be grouped into column families which contain data that will be accessed together. Retrieving two column families and joining them in the data access layer to perform a query will affect performance and should be avoided. This issue also applies

to other types of NoSQL data store. Schram and Anderson [126] design their data model for Cassandra manually, based on their understanding of the system. It is also noted that they do not move all of their data into Cassandra, but instead use Cassandra for new data while keeping the existing data in MySQL. However it is not noted how they keep track of where the data resides for the purpose of queries.

ExSchema

ExSchema is an approach for extracting database schemas through analysis of a software system's source code [127]. Castrejón et al. argue that the use of scalable and heterogeneous data stores within a system (polyglot persistence) is gradually becoming common practice, and that data reverse engineering tools must also change to accommodate this. As NoSQL databases are typically schemaless, the data schema for a system resides in the source code rather than the database.

A challenge emerges from the shift to schemaless databases, in that it is now more difficult to gain a high level overview of a systems data structures (which is valuable in understanding a system) [127]. This is the issue that ExSchema seeks to address. ExSchema currently supports the analysis of the native Java APIs for Neo4j, MongoDB, HBase, and CouchDB, as well as the analysis of relational databases where the Java Persistence API (JPA) has been used to handle the database operations.

Through ExSchema, Castrejón et al. set out to create a tool which would integrate seamlessly with existing development environments, yet would be simple to extend in order to add support for numerous different data stores. ExSchema has been implemented as an Eclipse plug-in [128] which is based on a common programming interface for NoSQL databases developed by Atzeni et al. [129]. The tool allows a developer to produce a diagram illustrating the database schemas in an application.

The Eclipse-based tool is composed of five main elements, a meta-layer and four source code analysers which gather information on the schema from different aspects of the code. The source code analysers are, a declarations analyser to gather information on variable declarations, a update analyser to discover code which performs update operations on the various databases, a repositories analyser to gather information on the history of the schema and how it has evolved by connecting to a Git repository, and an annotations analyser to gather information from relevant Java annotations. The meta-layer is conceptually above the code analysers and combines the information from all of the analysers to produce a PDF containing a diagram that illustrates the schema for each database.

While ExSchema is a very specific approach, it is still of interest due to the small number of tools/approaches currently available for extracting data models in this way. In particular the use of the common NoSQL interface developed by Atzeni et al. [129], has potential to be built upon. However, the key limitation of ExSchema is that it does not create a standard model which could be exported and used for other purposes.

Schemol

Schemol [4] is a domain specific language (DSL) tailored for extracting models out of 'web 2.0' databases. As part of this work the 'harvesting' of models from databases and facilitating migration to new platforms is analysed. Model harvesting is defined as the process of obtaining models from other software artefacts, such as code, databases, spreadsheets, etc. Díaz et al. identify web 2.0 databases as those which persist data for web 2.0 software, including blogs, wikis, and content management systems. In these types of system it is the data which is valuable, while the software can be changed to other commercial and open source platforms depending on requirements. However the migration of the data typically requires re-engineering due to the embedded mark-up and constructs which wrap the data as well as the different table/column structures. The embedded mark-up creates a large conceptual gap between the data and a target metamodel.

Díaz et al. [4] state that before Schemol, model harvesting could be performed in one of three ways. First, was the use of raw programming with direct database access. Next, was the use of object relational mappers. Finally, model-relational mappers could be combined with model-to-model transformations. These approaches are all lengthy and difficult, Schemol simplifies the model harvesting by providing a way of defining how the model elements can be obtained from an existing database.

Schemol aimed to address three challenges: expressing mapping in an intuitive way, exploiting the annotated mark-up embedded in the database, and ensuring the consistency between the database schema and metamodel (for example a table in the database should remain a 'first-class citizen' and be represented as a class in the model). It is based on the premise that a wide conceptual mismatch exists between the database schema and metamodel.

The Schemol DSL uses a rule-based approach, much like Gra2MoL [66]. Each rule specifies how an element of the source model is mapped to one or several elements of the target model. Rules are grouped into a transformation, which may also have a preamble. The syntax of the language is straightforward and each rule is made up of

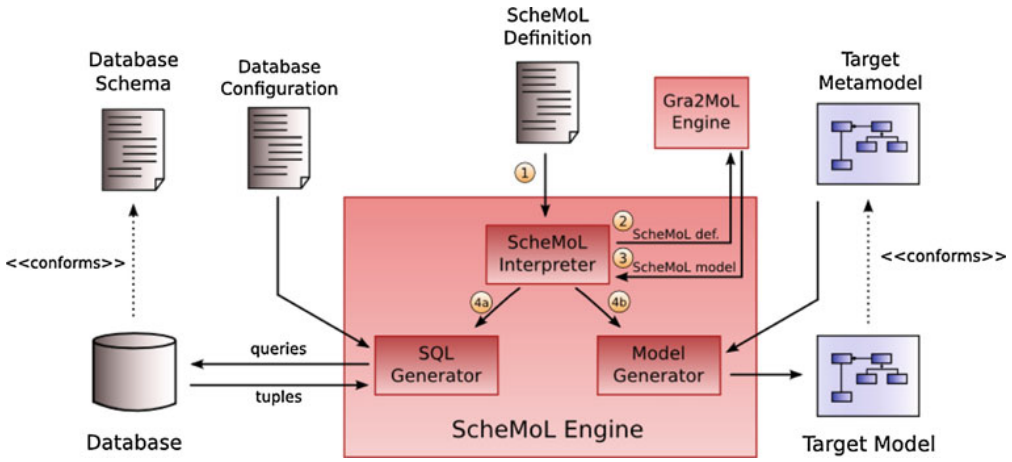

Figure 2.5: Schemol engine overview [4, p. 31]

four classes: from, to, filter, and mapping.

The implementation of the Schemol engine is based upon three components: the Schemol interpreter (which acts as a rule engine), the SQL generator, and a model generator. The execution order is identified by the numbered arrows in Figure 2.5. The engine was evaluated through a case study on a blogging system, with the Open Catalogue Format used as a target metamodel.

The ability of Schemol to express database-to-model mapping makes it relevant to the wider data re-engineering realm. The domain specific elements correspond to database-to-model mapping rather than interpretation of mark-up embedded in database item. The premise of the approach — that there is a large conceptual gap between the database schema and target metamodel — does not necessary hold when producing a model of a simple relational database. However, it does hold whenever a target metamodel defines a polyglot persistence data model.

2.5 Simulation

Research problems RP1 and RP2 focus on aspects of cost estimation for cloud database migration. However, the cloud migration case studies (Section 2.3.2) and existing methodologies (Section 2.3.2) show that this is a complex activity. A large number of stochastic variables are involved, such as database workload and cloud instance performance. As a result, this section reviews the leading cloud simulation approaches which could be used to address these research problems.

Several simulation tools and frameworks have been designed for cloud computing, such as: CloudSim [40], DCSim [5], SPECI [130], GreenCloud [131], and iCanCloud [132]. Many more cloud simulators extend CloudSim or use it as a discrete-event simulator, e.g., CloudAnalyst [133] and EMUSim [134]. This section presents the leading cloud simulation tools and methods which are suitable for migration simulation.

2.5.1 CloudSim

CloudSim [40] is a discrete-event simulation toolkit — written in Java — which enables the simulation and modelling of cloud compute systems. Rather than being an 'out-of-the-box' tool, CloudSim is designed to be extended and adapted for a wide variety of tasks requiring cloud simulation. It can simulate single, hybrid, and federations of clouds on a modest desktop PC.

The developers describe how successful cloud platforms require complex algorithms for provisioning, composition, configuration, and deployment; these are difficult to evaluate for several reasons. First, clouds have variable demands, supply patterns, system sizes, and resources (hardware, software, network). Their users have heterogeneous, dynamic, and competing Quality-of-Service requirements. Finally, they can be 'massive-scale' which makes evaluation (with multiple runs) time-consuming. This makes simulation the most viable solution to evaluate the performance of new algorithms or parameters in a repeatable manner.

CloudSim addresses several shortcomings in existing grid simulation tool-kits which restrict their use with clouds. Multi-layer service abstractions (i.e., SaaS, PaaS, and IaaS) are common in public clouds, and require additional modelling in a simulation. Support for cloud brokers and data-centre type environment is also needed.

CloudSim has a layered design: the Core layer acts as the discrete-event simulation engine, the CloudSim layer provides objects to model the datacenter environment, and the User layer provides objects to model Hosts, VMs, and cloud applications. The CloudSim and User layers also provide a number of interfaces to support the implementation of VM provisioning, cloud brokerage, and resource allocation.

The CloudSim framework has been used extensively to investigate a number of cloud computing problems, such as: energy-efficient datacenter management [135], fault-tolerance [136], and maintaining QoS requirements [137]. However, one key (deliberate) limitation is its lack of support for multi-threading and clusters [40]. It has been shown to handle simulations containing one million servers, although eventually some extreme-scale users will encounter scalability issues. Furthermore, the documentation

of the CloudSim codebase is limited which may cause problems when extending it.

NetworkCloudSim

NetworkCloudSim [138] extends CloudSim and adds: (1) a network flow model with support for bandwidth sharing and latency, and (2) objects representing MPI, parallel and multi-tier systems. The authors argue that current cloud simulation tools do not adequately support real networked cloud datacenters. Scientific applications, such as those for, climate modelling, drug design, and protein analysis are common cloud workloads. Accurate evaluation of algorithms affecting these types of application requires a simulation model that includes the datacenter network.

One shortcoming of CloudSim is the assumption that each Virtual Machine in the simulation is connected to all other Virtual Machines [138]. NetworkCloudSim addresses this by using a multi-layered network topology containing a root switch, multiple aggregate switches, and multiple edge switches. Virtual Machines can only be connected to edge switches, and all Internet traffic exits the datacenter through the root switch. The connections between all the switches and VMs have a defined bandwidth.

NetworkCloudSim provides AppCloudlet, NetworkCloudlet, Task, and Packet objects to model parallel applications. The AppCloudlet represents the whole application, while NetworkCloudlets represent smaller internal components. Each component consists of numerous Task objects, which can be in one of the following states: execute, receive, or send. All communication within the application takes place between the Tasks by sending Packets.

NetworkCloudSim was evaluated using a MPI application running on a small private cloud (four physical servers). The average execution time for the simulated and real-world applications was compared, and authors reported an error of less than one second. The MPI application generated random numbers in eight parallel processes and sent the values between these. While this evaluation was small-scale, the features of NetworkCloudSim have been successfully used in a number of different studies [139] [140] [141].

2.5.2 DCSim

The motivation behind DCSim is similar to CloudSim. Tighe et al. [5] argue that VM allocation and consolidation algorithms are a vital part of large-scale clouds. Thorough evaluation and experimentation with these algorithms is impractical in the real-world.

DCSim is a simulation framework which focuses on Infrastructure-as-a-Service (IaaS) clouds.

A DCSim simulation consists of a single datacenter with multiple hosts, a VM placement policy, and a Management policy. Each host has multiple VMs, a resource manager, a CPU scheduler, and a power model. A VM has an application and a description. The resources allocated to VMs are defined in the *VM Allocation object*. This structure is illustrated in Figure 2.6. Like CloudSim, it is a discrete-event simulator written in Java and is publicly available. Consideration is given to hypervisor loads as each host is required to contain a special Virtual Machine Manager VM.

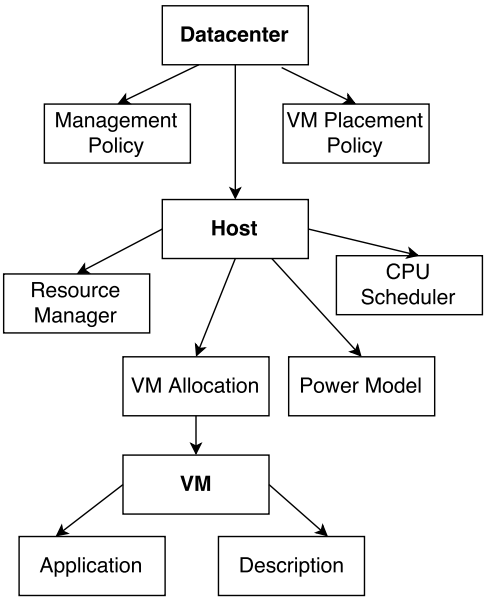

Figure 2.6: Key objects in DCSim, adapted from Figure 1 in [5].

DCSim provides additional features for power modelling compared to CloudSim. Each host as a power model defining its consumption at a given CPU utilisation. The authors have used real-world power consumption data from another study to calibrate these models. As the real-world data was measured in CPU utilisation intervals of 10% linear interpolation was used to estimate all values. Every host in the datacentre can be in one of the following power states: on, off, and suspended.

The CPU, RAM, bandwidth, and storage have individual resource managers to define how each is shared between VMs. By default the CPU is time-shared and all other components are space-shared. At the datacenter level DCSim requires that there is at least one VM placement policy which defines where a VM will be created

once requested. The default policy is to create a VM on the host with the highest utilisation which still has space for the specified VM. This policy can also migrate VMs to consolidate them, and allow hosts to be powered down.

The load on the VMs (and therefore the Hosts) is driven dynamically by the *Application* object. This can be implemented to apply any type of workload, although a default is provided which replicates the workload of a web server. Applications can also span multiple VMs.

The strengths of DCSim include: power modelling support and multi-VM applications with dynamic workloads. However, it lacks support for communication between applications, network modelling, and multiple datacenters. DCSim's evaluation shows longer execution times than CloudSim for the same number VMs and Hosts. The results also indicate the time complexity is greater than linear. Despite this Tighe et al. still performed simulations with 10,000 hosts and 20,000 VMs in less than one hour on a single desktop PC.

2.5.3 Interpolation and Extrapolation

Within a dataset interpolation and extrapolation methods estimate values for which there is no data. In their most simple form these methods can estimate value of a dependant variable Y corresponding to a given value of an independent variable X [142]. Interpolation is when X lies between two existing data points, and extrapolation is when X is outside the range of all existing data.

Using an example dataset of lines-of-code (Y) in a software system measured at fortnightly intervals (X). Extrapolation is performed when the development team want to know many lines-of-code the system will contain three months in the future if current progress is maintained. Interpolation is performed when the development team wish to estimate the system's lines-of-code at week 15 when measurements only exist for week 14 and week 16. Interpolation and extrapolation methods are typically used during the analysis of empirical results, but also when constructing models for a simulation (e.g., in [143] and [134]).

The assumptions and accuracy of these methods are discussed extensively in the literature [142]. Typically, it is required that no violent fluctuations occur in the dataset and that the change in the two variables is governed by some rules i.e., Y is a function of X. It should be noted that extrapolation is inherently biased and care must be taken when extending far beyond the last measurement or when a large range is expected in the projected values [144].

The following subsections present some common interpolation and extrapolation methods.

Ordinary Least Squares

Ordinary Least Squares (OLS) is the most straightforward form of regression analysis [142] (a statistical technique for modelling the relationship between variables [145]). It is a method to determine the 'best' linear approximation of the relationship between the independent and dependant variables.

Consider some data from the previous example (lines-of-code in a software system) plotted on a scatter graph, as shown in Figure 2.7. The 'best' regression line is the one which minimises the error between the actual data points and the line. This line, which is defined as a function, can then be used to perform interpolation and extrapolation. In Figure 2.7 the following data was used: $x = 0, 2, 4, 6, 8, 10, 12, 14, 16$ and $y = 0, 241, 289, 1253, 1554, 1578, 1819, 2272, 2995$. The equation for the regression line (obtained with OLS) is: $y = 179x + (-97)$.

In the OLS method the vertical distance between each data point and a candidate regression is measured, then an average of all these distances is calculated [146]. The regression line with the lowest (i.e., least) average distance is considered to the best fit for the data. In OLS this average is the sum of the squared values of the vertical distance. This can be formalised as the regression line which minimises [146]:

$$SSE = \sum_{i=1}^{N} (C_i - \hat{C}_i)^2$$

Where SSE is the sum of the squared errors, C is the data point, and $\hat{C}$ is the value that would be estimated by the regression line.

OLS is one of the most popular linear regression techniques (and many software libraries support it), although it has limitations which users must be aware of. In [147] it is argued that indiscriminate use of OLS is a problem in some research; specifically outliers and dependencies between the independent variables will affect accuracy. The biggest limitation of OLS, and any form of linear regression, is that often systems are non-linear [148]. It should therefore only be applied if linearity has been determined.

Nonlinear Least Squares

Nonlinear Least Squares methods perform the same task as Ordinary Least Squares, i.e., to determine the 'best' approximation of the relationship between dependant and

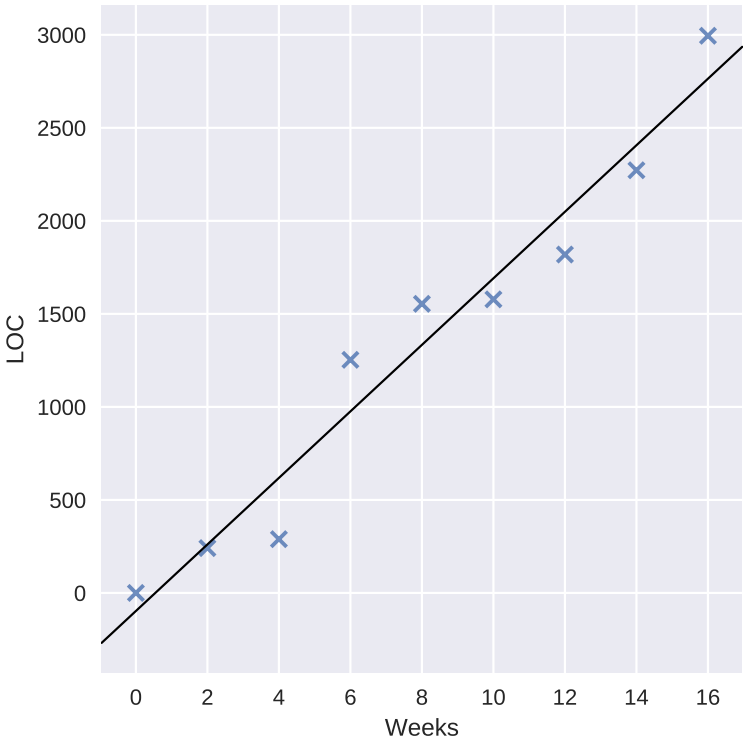

Figure 2.7: Example application of the Ordinary Least Squares method

independent variables. However, they support datasets which cannot be accurately modelled by a linear function [149]. This comes at a cost and there are a number of limitations a user must be aware of.

Nonlinear least squares methods cannot provide an exact solution, and require iterative methods to find the best fit [150]. As a result non-convergence is common (i.e., where no single result is found). A starting/initial value for the equation parameters must also be provided. These issues mean it is undesirable to use Nonlinear Least Squares methods unless non-linearity has been determined.

Numerous algorithms exist to perform non-linear least squares, typically these have different performance characteristics or different probabilities of non-convergence [151]. Like Ordinary least squares, most numerical analysis software packages implement a Nonlinear Least Squares method [150].

2.6 Chapter Summary

This chapter reviewed the field of cloud relational database migration. The migration of relational databases to — or between — clouds is typically performed as part of: complete system migration or legacy system modernisation. As a result, this chapter started by discussing leading legacy system modernisation approaches. Following this, cloud migration approaches were discussed. A common theme in these areas is the use of MDE (Section 2.2) and simulation (Section 2.5). MDE is employed to handle system complexity and increase automation. While simulation enables evaluation of methods, parameters, and algorithms which are impractical to evaluate in the real-world.

The leading legacy system cloud migration approaches are: REMICS and ARTIST. Both are tool-supported and focus on the software-layer. ARTIST provides more sophisticated feasibility and costing tools, business modelling tools (for software product to SaaS migration), and a different overall methodology. Several database layer re-engineering and migration techniques were identified to: extract schemas for legacy databases, map schemas to higher level models, and to migrate data. None of the published techniques uncovered focused on polyglot persistence migration.

The key challenges when migrating a system to the cloud have been identified. These include: security, costing, charging, SLA definition, heterogeneous APIs (vendor lock-in), determining suitability for the cloud, and reluctance among end users. Log2cloud presents a novel approach for determining the number of cloud instances required. CloudML is a DSL which allows the definition of cloud services. The only model-based migration approach identified was CloudMIG.

Architecture-Driven Modernisation is a discipline which studies the model-based modernisation of legacy systems. It has been spearheaded by the Object Management Group — through their Architecture-Driven Modernisation Task Force — who have proposed several standardised metamodels. Two of these metamodels are relevant to the research problems addressed in this thesis: the Knowledge Discovery Metamodel (KDM) and the Structured Metrics Metamodel (SMM). The KDM allows a legacy system (and its database) to be modelled from several perspectives and at different abstraction levels. The SMM allows measures to be defined and provides a standardised interchange format for measurements.

In the final section of this literature review CloudSim, DCSim, and Interpolation/Extrapolation approaches were analysed. CloudSim is an established discrete-event simulator for cloud computing platforms, and supports the evaluation of provisioning and resource sharing algorithms. It was extended with NetworkCloudSim

to consider network latency and bandwidth. DCSim is a competing simulator which provides support for power modelling and dynamic application workloads, but without networking modelling.

Chapter 3

Database Modelling

3.1 Introduction

Enterprise systems typically reach a point in their life cycle when their database layer must be modified, as discussed in Section 2.4. This can include: increasing capacity, moving to a new platform, archiving old data, or refactoring. These activities require a detailed understanding of the database structure and workload. However, the existing database modelling and model-based re-engineering approaches have limited system support (Section 2.4.2). Each also focuses on a single aspect of modelling or re-engineering, hence an organisation would require a disparate suite of approaches to obtain a complete model of the database. The problem is exacerbated when non-standard metamodels are used, or the model extraction and consumption steps are inseparable.

This chapter presents DBLModeller, a tool-supported approach for extracting structure and workload models from a system's database. These conform to the Knowledge Discovery Metamodel [35] and Structured Metrics Metamodel respectively [2], which were identified as leading standards for model-based re-engineering in Chapter 2. It is platform and system agnostic and provides organisations with sufficient information to perform effective database re-engineering or migration (e.g., to a cloud platform). DBLModeller addresses research problem 1 and 2 from Section 1.2. A preliminary version of the approach was published in [6].

DBLModeller shortens the model extraction process by removing the need for intermediate models and transformations. This is possible due to the novel use of annotated text-to-model transformation, and reduces the effort needed to extend DBLModeller. Secondly, DBLModeller uses a multi-dialect SQL grammar which supports common

constructs. An empirical study was performed as part or this research project which identified commonality in database schemas from several real-world systems. The heterogeneity of databases, schemas, and systems is a key reason why the existing work (e.g., [39]) supports a limited subset of SQL for one database type. As a result tailored support and easy extensibility are key advantages of DBLModeller.

The DBLModeller approach was implemented as a tool to enable its evaluation, which is available online [9]. It was compared with the state-of-the-art Gra2MoL SQL-to-KDM tool [39] from multiple perspectives: extensibility, system support, model completeness, model correctness, and performance. A case study was used to evaluate extensibility where both DBLModeller and Gra2MoL were modified to support a Microsoft Sharepoint database schema. The specialisation/adaption of these tools for a specific schema represents an extension to support an 'edge case' system that uses uncommon database constructs.

Performance was evaluated by comparing model extraction times from four real systems. One was the closed-source Science Warehouse system (a procurement/finance system), two were popular open-source business applications, and last was the education system used by the Gra2MoL evaluation [39]. This provided a selection of different database sizes (SQL schemas between 400-31,500 lines-of-code) from different domains. Model completeness and correctness were measured by comparing the input and output elements when extracting models from these four systems. Compliance with the underlining metamodels (i.e., KDM and SMM) was also checked.

System support was evaluated by performing a keyword analysis on 18 database schema dumps. This compared the SQL keywords supported the DBLModeller and Gra2MoL tools. Additionally, the most commonly occurring keywords were identified and their support in DBLModeller was checked. The impact of any unsupported keywords was considered, i.e., whether the output model is affected and how.

The results from these evaluations show that DBLModeller has significantly improved extensibility and SQL support, and the models extracted in less time. Extending DBLModeller to support the Sharepoint schema required fewer changes and over 50% less code. The keyword analysis showed that DBLModeller supported between 96% and 99% of the content the sampled SQL schemas. DBLModelled also extracted models in less time for every schema studied, with a 86% improvement in the best case.

The rest of the chapter is structured as follows. Section 3.2 starts by outlining the key steps in the DBLModeller approach, along with their inputs and outputs. Next, the novel method used for text-to-model transformation is explained and compared with Gra2MoL. The model-refinement method is presented as an algorithm, and its time

complexity is analysed. Section 3.3 describes in detail how DBLModeller is evaluated and presents the results. The threats to the evaluation's validity are also discussed. Section 3.4 summarises the chapter by highlighting the key results and contributions.

3.2 DBLModeller Approach

DBLModeller is a three-step approach for extracting database structure models (conforming to KDM) and workload models (conforming to SMM). In Step 1, two inputs are obtained from the system: the database schema, and an SQL trace of the database traffic. The schema should be extracted from the live database, as existing scripts for generating it may be out of date. The SQL trace (i.e. the queries sent to the database) is then processed into a sequence of load measurements, these are: entity count, entity reads, entity writes, unused entities, and database size. An 'entity' is the primary data artefact the database is storing; e.g. products in an e-commerce system or pages in a Wiki. Abstracting the measurements in this way greatly simplifies the workload model extraction, and makes the process more generic.

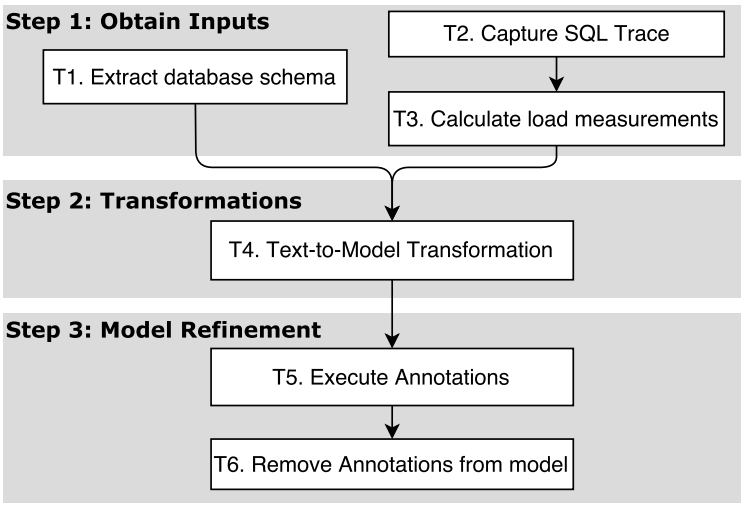

Figure 3.1: Overview of the DBLModeller approach

Existing tools for obtaining the inputs are commonly available. For example, Oracle SQL Developer (an IDE for Oracle databases) provides an export wizard to produce an SQL file containing the schema. Platform-independent alternatives also exist, such as [152] and [153]. Similarly, the SQL trace can be captured using a range of propertyless (e.g., Oracle JDBC logging) or platform independent tools (e.g., P6Spy). These tools run on the application servers and save SQL queries to a logging file or service. The

performance impact of this can vary according to the database, type, size, and workload. It is recommended that organisations measure the performance impact and determine whether it is acceptable.

Step 2 uses text-to-model transformations (T2M) to create annotated KDM and SMM models from the schema and workload measurements. These are valid models (conforming to the KDM [51] and SMM [2]), where textual annotations have been inserted into free-text fields of specific model elements. Any existing tool that supports these metamodels can work with the models in this state, although the annotations will be ignored. There are three types of annotation (as discussed later in Section 3.2.3) which all manipulate their host model to increase its detail. They are automatically processed and executed in Step 3.

The annotations and Step 2 are a direct replacement for a model-to-model transformation in previous approaches [66]. Their role is to accommodate the transition from relatively low-level SQL to high-level abstract models (i.e., KDM and SMM). For example, separate statements are typically used in SQL schema dumps for creating tables and adding indexes to tables. In [66] a T2M and M2M were used to combine these into a single model element. While in DBLmodeller an annotated KDM model is created with an orphaned 'IndexKey' model element containing a MOVE annotation. When the annotation is executed, this is moved so that it is nested within the correct table model element.

A second task which requires Step 2 is the creation of the relationship between a field and its data type. In a KDM-based model the 'data' package must be used to model the database, while the 'code' package must be used to define its data types. When a column is found within a SQL 'create table' statement, a T2M transformation rule creates a KDM 'itemUnit' element without a type. The T2M transformation rule also adds an annotation to the itemUnit. Once the annotation is resolved, the model element will refer to the correct type in the KDM code package. The datatypes present in a particular SQL schema will typically be an unknown subset of those supported by the SQL dialect.

The goal of replacing a M2M transformation with annotations is to improve performance and extensibility. This is discussed and evaluated in the later sections.

3.2.1 T2M Transformations

DBLModeller uses grammar-to-model mapping [66] to perform the T2M transformations. This involves in the mapping of a grammar (defining the source text) to a

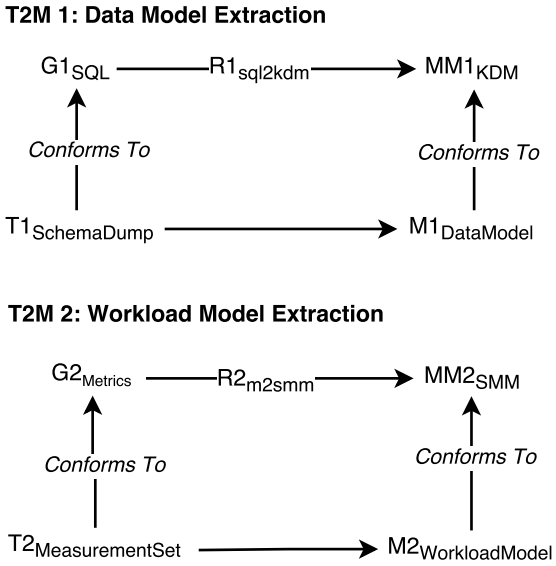

Figure 3.2: Text-to-model transformations in DBLModeller

metamodel (defining the target model), with a domain-specific language specifying the mapping rules. However, the standard process has been extended in DBLModeller with an annotation processing engine.

Grammar-to-model mapping is a generic solution extracting models from source code, which required less impleation effort compared to alternatives [66]. It was selected as the base for DBLModellers transformations as a result. Furthermore, the approach authors have used it successfully for database layer modernisation [39]. In this work, a KDM-based model is extracted from a PL/SQL application. This model is then analysed to establish the coupling between PL/SQL triggers and the user interface. It is this work that DBLModeller is compared to in the evaluation.

Typically when following a grammar-to-model mapping approach it is beneficial to re-use existing publicly available grammars to reduce the development effort [66]. However, when developing DBLModeller it became clear that this rule-of-thumb does not apply when languages with multiple dialects must be supported. As a result a dialect-independent grammar has been developed as part of DBLModeller ($G1_{SQL}$ in Figure 3.2).

In Figure 3.2, each rule in $R1_{sql2kdm}$ maps a grammar rule in $G1_{SQL}$ to a model element in $MM1_{KDM}$. Using an additional grammar for every SQL dialect would also require an additional set of mapping rules, resulting in a substantial amount of work. As extracting a KDM-based structure model involves a significant amount of abstraction

from the SQL DDL, it is straightforward to use a single grammar for multiple SQL dialects. Many SQL statements variations and data types become equivalent in the KDM, e.g. TEXT (in MySQL) and VARCHAR (in Oracle) both map to the KDM type String.

Annotations are inserted via the mapping rules in $R1_{sql2kdm}$ and $R2_{m2smm}$ by appending them to the *name* of a model element. Every model element in KDM and SMM has a *name* property, allowing annotations to be used anywhere.

The textual input to the workload model transformation $T2_{MeasurementSet}$ is a set of measurements rather than a language, also precluding the reuse of an existing publicly available grammar. The elements of this set are of the form:

$$M_n = (Name, TS_{Start}, TS_{End}, E_{Total}, E_{Reads}, E_{Writes}, E_{Unused}, DB_{Size}), \tag{3.1}$$

where the string $Name$ identifies a particular measurement (i.e. data point) in the model; $[TS_{Start}, TS_{End}]$ is the time interval when the measurements were taken; E_{Total}, E_{Reads}, E_{Writes} and E_{Unused} specify the number of primary database identities at the end of the time interval, and the numbers of reads, writes (i.e. additions, deletions and updates) and unused entities, respectively; and DB_{Size} is the size of the database. Note that in order to obtain E_{Unused}, the query logs (or an equivalent data set) must be compared to the schema dump.

The metrics from (3.1) were chosen because they affect cloud migration and running costs, and therefore are particularly relevant this research. Multiple measurement periods of any length are permitted to ensure that the precision of the extracted workload model matches the available data or requirements.

3.2.2 KDM Compliance

The Knowledge Discovery Metamodel has a large scope, with twelve packages distributed across four abstraction layers. Each package allows some legacy system artefact to be modelled. The metamodel's developers — the Object Management Group — proposed compliance levels [35] for tools using the metamodel, this reflects the fact that many tools (such as DBLModeller) do not wish to implement all of the KDM. Level 0 complaint tools support the: core, kdm, source, code, and action packages. A level 1 tool adds support for one or more of the: platform, data, event, UI, build, structure, and conceptual packages. Finally, level 2 tools support the entire metamodel.

The DBLModeller tool and approach only support the KDM data package and none of the parent packages. As a result it does not comply to any of the levels.

However, this is a deliberate design decision made during this work. The KDM and the current tools and approaches which implement it, have a different purpose/goal compared to DBLModeller. The goal of DBLModeller is to provide a vendor-neutral database modelling approach that captures cloud-specific properties. Whereas the KDM is intended to provide standardised representation covering every artefact of legacy software system.

The key difference between the work described in this thesis and KDM is scope (i.e., database layer vs an entire system). DBLModeller is intended to be used within other approaches (e.g., [25] or [26]) where the present model extraction tools only support the software layer [53]. Furthermore, the lack of full level compliance does not affect viewing, modifying, and querying the model with the Eclipse Modelling Framework. There is also no impact on tools which only consume the data package (e.g., MigSim, which is presented in Chapter 4).

3.2.3 Model Refinement

During model refinement, the annotated KDM and SMM models are searched to locate annotations, which are then executed and removed. The three types of annotations used by DBLModeller are described below.

Move annotations act on the line on which they are placed, moving it to within another element in the model. The target element is identified by its name and type, which are included as parameters in the annotation. With the standard Gra2MoL framework, the element order in the model will match the input text. This means that while 'CREATE INDEX' statements can be mapped to KDM:IndexElement elements they cannot be placed within the correct KDM:RelationalTable, hence the Move annotation is needed.

Add annotations create new model elements in a specific model location, this can either be the same location as the annotation or a different one. This is used to handle 'ALTER TABLE' statements and keys within 'CREATE TABLE' statements, as both would be impossible with a single Gra2MoL T2M transformation. For example, processing an ALTER TABLE statement requires some existing model element to be modified (e.g. add a primary/key foreign relationship). This goes against the traditional function of a T2M transformation which creates a model element when a statement is found in the text.

Reference annotations serialise name-based references (e.g. SchemaName. TableName) to model paths (e.g. $//@element.1//element.3//element.0$). Though Gra2MoL

has a mechanism for tasks like this, an annotation is needed because model elements can be created after the grammar-to-model mapping is complete.

Algorithm and Performance

The DBLModeller model refinement is formalised by function RefineModel from Algorithm 1. Given an *annotatedModel*, this function first invokes GetAnnotations to obtain a list of Move annotations that is passed to ExecuteMoves (line 2). This ensures that all Move annotations, which will change the location of the other annotations, are executed before obtaining the Add and Reference annotations and passing them to ExecuteNonMoves (line 3). Note that the *annotatedModel* is the (text) output of the text-to-model transformation from Step 2 of DBLModeller, and is processed as a sequence of lines. Its size grows linearly with the size of the schema dump or load measurement set used to create it (typically up to a few megabytes for a single system).

To analyse the time complexity of the algorithm, it is noted that the GetAnnotations for loop from lines 7–11 iterates over each model line to check if it contains an annotation, therefore taking $\mathcal{O}(n)$ time for an n-line *annotatedModel*. ExecuteMoves iterates over each of the at most n Move annotation in order to: remove the target line from the model ($\mathcal{O}(1)$ time); perform a linear search of the model to find the line number of the new location ($\mathcal{O}(n)$ time); and insert the target line ($\mathcal{O}(1)$ time). The complexity of ExecuteMoves is therefore $\mathcal{O}(n^2)$. Finally, executeNonMoves iterates over each Add and Reference annotation (lines 16–20), performing an $\mathcal{O}(n)$ linear search followed by an $\mathcal{O}(1)$ insertion or an $\mathcal{O}(n)$ replacement of a model line. As the model may contain up to $\mathcal{O}(n)$ Add and Reference annotations, the overall complexity of ExecuteNonMoves, and thus of the entire model refinement step of DBLModeller, is $\mathcal{O}(n^2)$.

DBLModeller has been implelmented in Java and made publically available on the project webpage [9]. The current version supports MySQL and Oracle databases, which together hold a substantial market share [154, 155].

The recommend approach for obtaining the SQL dump is via MySQL Workbench [156] or Oracle SQL Developer [157], both of which ensure that the dump is accurate and in a recognised format. The tool supports the statements listed in Table 3.1. DROP statements (their role in database scripts is purely to clean the database before executing CREATE statements) and comments in the SQL (i.e. /*text*/ or //text) are ignored as they do not affect the KDM model.

Algorithm 1 Model refinement

```
1: function RefineModel(annotatedModel)
2:     executeMoves(annotatedModel, getAnnotations(annotatedModel, {MOVE}))
3:     executeNonMoves(annotatedModel, getAnnotations(annotatedModel, {ADD, REF}))
4: end function
5:
6: function GetAnnotations(annotatedModel, types)
7:     for i ← 0 to size(annotatedModel) do
8:         if annotatedModel.line(i).type ∈ types then
9:             list.add(i, annotatedModel.line(i))
10:        end if
11:    end for
12:    return list
13: end function
14:
15: function ExecuteMoves(annotatedModel, annotations)
16:    for each annotation a in annotations do
17:        source ← annotatedModel.remove(a.lineNo)
18:        lineNo ← search(annotatedModel, getTargetId(a))
19:        annotatedModel.insert(lineNo, source)
20:    end for
21: end function
22:
23: function ExecuteNonMoves(annotatedModel, annotations)
24:    for each annotation a in annotations do
25:        lineNo ← search(annotatedModel, getTargetId(a))
26:        if a.isAdd then
27:            annotatedModel.insert(lineNo, a)
28:        elseif a.isReference then
29:            annotatedModel.replace(a.LineNo, resolveReference(lineNo, a))
30:        end if
31:    end for
32: end function
```

Table 3.1: Supported SQL constructs

Statements	**Included Parameters**
CREATE TABLE	Key, Primary Key, Unique Key, Default column values, Non-null columns
CREATE INDEX	Unique
CREATE SEQUENCE	Minimum, Maximum, & Start value, Increment by
ALTER TABLE	Foreign Key, Primary Key
COMMENT ON	Table, Column

The SMM-based model of the database workload is produced from Oracle JDBC query logs [158], via a CSV file formatted as shown in Table 3.2. While DBLModeller can only generate the CSV from Oracle query logs, it would be possible to extend it for MySQL [159]. Using a CSV file as input for the workload model T2M transformation allows other system artefacts or data to be used as input (e.g. event logs or monitoring data) without code changes to DBLModeller. When transforming Oracle query logs some manual input is still required to specify the metadata fields (System Name, Scope, and Entity Name) and those requiring database access (Unused Entities and Database Size).

The implementation of the T2M transformation uses the Gra2MoL tool [160], which required the development of a CSV grammar, CSV-to-SMM mapping rules, an SQL grammar, and SQL-to-KDM mapping rules. Ecore/EMOF implementations of the KDM and the SMM were already available [52, 161]. The annotations below are inserted as specified by the CSV-SMM and SQL-to-KDM mapping rules, and are then executed by DBLModeller's Annotation processing engine:

- `;DBLM_MOVE: targetElementName, targetElementType;`
- `;DBLM_REF: referenceAttributeName, referenceTargetName, referenceTargetType`
- `;DBLM_ADDH: name, type, xsitype, referenceAttributeName1, referenceTargetName1, referenceAttributeName2, referenceTargetName2, referenceTargetType;`
- `;DBLM_ADD: targetElementName, targetElementType,*; ;POSTPROCESS_ADD :name, type, xsitype, referenceAttributeName, referenceTargetName, referenceTargetType;`

Note that two types of Add annotation exist in the DBLModeller implementation: one to support the creation of model elements at the current location (ADDH) and one that creates model elements at a specified location (ADD). Furthermore, the ADD statement contains two parts to allow the addition of multiple model elements in the same location.

Table 3.2: CSV file format

Line	Data
1	System Name (String), Scope (String), Entity Name (String)
2 .. n	Measurement Period Alias (String), Measurement Period Start (Timestamp; yyyy-mm-dd hh:mm:ss), Measurement Period End (Timestamp; yyyy-mm-dd hh:mm:ss), Entity Count (Integer), Entity Reads (Integer), Entity Writes (Integer), Unused Entities (Integer), Database Size (Double)

3.2.4 Query Logging Performance Impact

A key step in the DBLModeller approach is the use query logs / SQL traces to model the database workload. A query log contains the SQL queries sent from the application to its database, along with a timestamp. This provides a record of the type and quantity of data being inserted, updated, or accessed. It can be used for: debugging queries, determining 'hot spots' [162] in the database and identify whether the database or its tables are read or write heavy. However, in this research project it used to determine hourly, daily, weekly, or monthly load patterns, as well as growth the query volumes and database size. This information may be unknown or misunderstood for some systems, especially those considered legacy [163].

Obtaining query logs with minimal performance impact is a non-trivial research and engineering problem for database vendors and system developers. As a result several approaches exists, including: proprietary database features (e.g., MySQL General Query Log [159]), proprietary libraries (e.g., Oracle JDBC logging [158]), and multi-platform interceptor tools (e.g., P6Spy [164]). To determine the feasibility of these techniques in a production environment its necessary to the performance impact, which must be minimal. It is expected that these approaches will vary according to the efficiency of their implementation.

Using the Apache OFBiz ERP system [165] and a set of publically available performance tests [166], the performance impact of three query logging techniques has been measured. The performance tests perform data-intensive activities (e.g., adding new customers, making orders, and altering warehouse inventory). For each query logging approach the performance test suite was executed 10 times to gather accurate results. The results are presented in Table 3.3.

Table 3.3: OFBiz performance test results for query logging frameworks

Logging Framework	Mean Request Time (ms)	Std Dev.
No Logging	114	13
Oracle [158]	5279	228
Log4jdbc [167]	7640	292
P6Spy [164]	149	23

The best performing approach was P6Spy (a JDBC interceptor tool) which reduced performance by approximately 30% in this case. While the Oracle and Log4jdbc approaches drastically reduced performance. This result is primarily due to Oracle and Log4jdbc using exceptions to determine where the query originated from the system (i.e., by proving a stack trace). While this information is valuable for debugging it is not used by DBLModeller. Furthermore, P6Spy uses of multi-threading and caching to further improve performance.

The results in Table 3.3 are heavily system and workload dependant, however P6Spy's more sophisticated implementation is likely to offer the smallest performance impact in most cases. It is expected that the performance degradation can be mitigated or justified in order to capture an accurate query log. As a result the DBLModeller tool supports query logs in the P6Spy and Oracle format.

3.3 Evaluation

The evaluation of DBLModeller aimed to answer the following research questions:

RQ1 How does the set of SQL keywords supported by DBLModeller compare to those used in Oracle/MySQL dumps from real-world systems?

RQ2 What is the effect of a multi-dialect grammar and annotated T2M transformation on extensibility?

RQ3 How is the model extraction time impacted by the M2M transformation replacement?

RQ4 To what extent are the models produced by DBLModeller complete and correct?

RQ1 was answered by extracting and analysing 18 schema dumps from 15 systems, as discussed in Section 3.3.1. RQ2 was answered through a case study which adapts DBLModeller to support a Microsoft SQL Server based application (Section 3.3.2). Finally, RQ3 and RQ4 were answered empirically by extracting models from a series of different systems. The data used in this section is available on the project website.

3.3.1 SQL Keyword Usage Study

Fully supporting every SQL dialect was impractical due to the number that exist and the size of the language, therefore DBLModeller supports a subset of two dialects (Oracle and MySQL). Whilst it is straightforward to identify which dialects to support (many organisations report on the estimated market share), it is harder to select statements and keywords to support within these. The generality of the DBLModeller tool will be greatly affected by this decision, making RQ1 particularly important.

Table 3.4: Database schemas used for keyword analysis

System	**Type**	**Domain**
Science Warehouse	Oracle	E-commerce
University of Murcia Record System	Oracle	Record System
Apache OFBiz	Oracle & MySQL	Business Management & E-commerce
MediaWiki	Oracle & MySQL	Collaboration
Confluence	Oracle & MySQL	Collaboration
Joomla	MySQL	Website Management
Magneto	MySQL	E-commerce
SonarQube	MySQL	Software Engineering
Mantis	MySQL	Software Engineering
WordPress	MySQL	Website Management
Moodle	MySQL	Education
OrangeHRM	MySQL	Record System
SuiteCRM	MySQL	Business Management
RefBase	MySQL	Education
OpenMRS	MySQL	Record System

The evaluation of RQ1 requires: a list of Oracle and MySQL keywords [168] [169], a list of those supported in DBLModeller, and schema (DDL) dumps from multiple Oracle and MySQL based systems. Schema dumps have been obtained from 15 systems as listed in Table 3.4; it was possible to obtain both Oracle and MySQL schemas for OFBiz, MediaWiki, and confluence. Each system was installed and started, then MySQL Workbench or Oracle SQL Developer was used to connect to the database and produce the schema dump. The exception was the Science Warehouse system, as this is a live system it was possible to obtain a schema dump from the production database.

DROP statements and comments were removed, then the occurrences of each

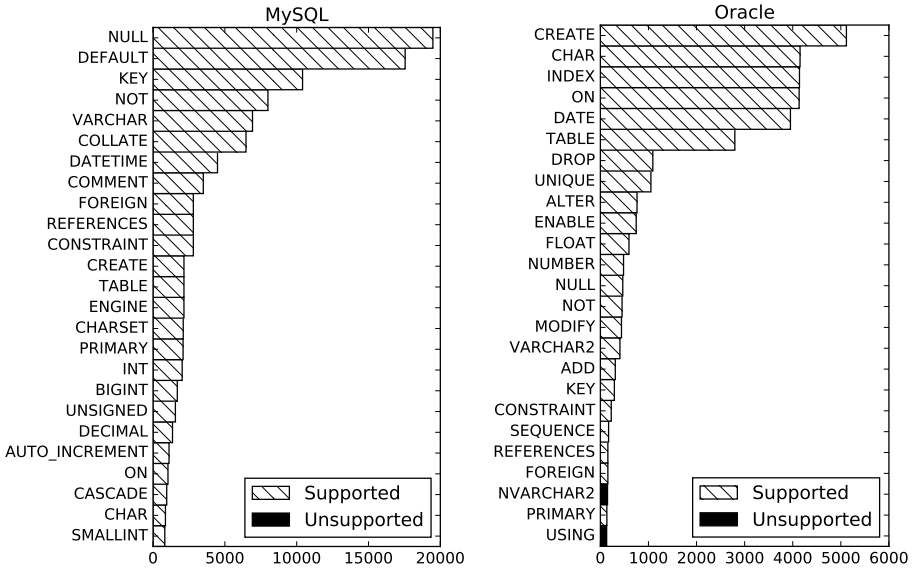

Figure 3.3: Most frequent SQL keywords for the MySQL (left) and Oracle dialects (right)

MySQL and Oracle keyword in their respective schema dump files was counted. As mentioned previously, DROP statements in schema dumps do not affect the exported model due to their role. It was expected that most systems would make heavy use of a small subset of permissible keywords, with outliers (i.e. rare keywords) in some systems.

The following results were obtained:

- Only 78 MySQL keywords were used in the 13 MySQL schema dumps;
- Only 58 Oracle keywords were used in the 5 Oracle schema dumps;
- This is only 16% of the permissible keywords for both databases (78/492 and 58/370).

As shown in Figure 3.3 the keyword usage is long-tailed, with the top 5 keywords having more occurrences than the rest of the keywords combined. In order to support the maximum number of systems, DBLModeller's supported subset of SQL should align with the subset of frequently used keywords. Hence both figures also show unsupported keywords.

None of the words which appear in the MySQL top 25 are unsupported, while only two in the Oracle top 25 are unsupported: NVARCHAR2 and USING. NVARCHAR2

is only used by Confluence (albeit extensively), however this means the data type for NVARCHAR2 columns will be null. If necessary, support for NVARCHAR2 could be added by modifying one line of the SQL grammar (to map it to the KDM:String data type). The lack of support for USING is not an issue because it specifies whether an index is enabled or disabled in the confluence schema, and this detail is lost when abstracting to a KDM model.

When an unsupported keyword is encountered by DBLModeller an error message is produced during the text-to-model transformation. The error message will be produced by the ANTLR framework which is used to parse the text during this transformation. If the unsupported keyword is used infrequently and is not critical to the statements in the schema, then DBLModeller will continue and produce a partial model.

Given the original research question (How do the SQL keywords supported by DBLModeller compare to those used in dumps from real-world systems?) it can be concluded that the majority of frequently used keywords are supported. The keyword analysis revealed that DBLModeller supported 96% of the content in the MySQL schemas and 99% of the content in the Oracle Schemas.

3.3.2 Microsoft SQL Server Specialisation

Given the heterogeneity of schemas, databases, and SQL dialects, it is inevitable that DBLModeller may need to be extended. In this section RQ2 (What is the effect of a multi-dialect grammar and annotated T2M transformation on extensibility?) will be answered. A case study has been performed where DBLModeller and the Gra2MoL SQL-to-KDM extraction tool [39] [160] were extended to support a Microsoft SharePoint schema.

The schema dump was obtained from a Microsoft SQL Server database used by a Microsoft SharePoint 2013 instance. This instance was created specifically for the case study by installing SharePoint. The schema (rather than the database content) is needed here, so the results are not affected because the SharePoint instance is not live/in-use by an organisation. As SharePoint uses 16 schemas the largest was selected; this contains 7 KLOC consisting of 136 tables, 5442 columns, and 61 indexes. The goal here was not to select a schema which is representative of all Microsoft SQL Server based databases, but rather to have a schema which is unsupported by both tools in equal measure.

The changes needed to DBLModeller to support the schema were determined by attempting to extract a model, then noting any errors produced. These were then fixed

Table 3.5: Code changes to support the SharePoint schema

	G2M New Grammar		**DBLModeller Extension**	
	New LOC	Updated LOC	New LOC	Updated LOC
Lexer	70	0	25	6
Parser	11	0	6	14
T2M	25	85	24	1
M2M	10	5	0	0
Total	*116*	*90*	*55*	*21*

incrementally and the number of modified lines of code were counted. However, with Gra2MoL a new ANTLR grammar was developed to parse the schema dump. This makes it possible to compare the work required to extend a grammar against the work required to develop a new grammar.

Table 3.5 compares the new and updated lines of code (LOC) for the two approaches. The results are presented on a per-component basis to show where each change was made. The G2M (Grammar-to-Model) approach requires four components: a lexer, parser, text-to-model transformation, and a model-to-model transformation. As discussed previously, the DBLModeller approach improves upon this by removing the model-to-model transformation. This results in an immediate reduction of 10 new LOC and 5 updated LOC. Furthermore, the total new LOC is 54% lower for DBLModeller and the total updated LOC is 77% lower, confirming the code has not been moved elsewhere.

G2M exclusively required new lines of code for the lexer and parser, i.e., none were updated, whereas DBLModeller required a mix of new and updated lines of code. This is due to G2M requiring an ATLR grammar to be written if one is unavailable for the target SQL dialect. Both tools use identical technologies which ensures comparable results, specifically, ANTLR for the Lexer/Parser and the G2M DSL for the transformations. Returning to RQ2, it is concluded that the changes made in DBLModeller have had a positive effect on extensibility. The similarities between SQL dialects ensures extension is a straightforward task.

3.3.3 Model Extraction

This section evaluates model extraction performance (RQ3), along with the completeness and correctness of the resulting models (RQ4). These research questions have been examined together due to the impact of completeness on performance. It is expected that the removal of the M2M transformation from the model extraction process will

Table 3.6: Model extraction duration using DBLModeller and Gra2MoL

Schema	**KLOC**	**Tool**	**Mean (Secs.)**	**Std. Dev.**	**sec / KLOC**
Oracle OFBiz	31.5	DBLM	174	2.35	6
	10.3	G2M	237	3.4	24
Oracle MediaWiki	2	DBLM	7	0.23	4
	0.8	G2M	14	0.68	18
Oracle Science Warehouse	1	DBLM	5	0.19	5
	0.4	G2M	14	0.62	35
Oracle UoM Student System	0.3	DBLM	5	0.21	17
	0.3	G2M	10	0.62	33
MySQL OFBiz	21.7	DBLM	104	2.13	5
	9.5	G2M	230	9.73	24
MySQL MediaWiki	1	DBLM	5	0.24	5
	0.4	G2M	13	0.53	33

have significant performance gains.

DBLModeller has again been compared to Gra2MoL's PLSQL2KDM example [160] as this had the highest level of SQL support at the time of writing. A KDM model was extracted from the database schemas obtained in Section 3.3.1. This was repeated 20 times per schema and the execution time was measured. A virtual machine with 4GB of RAM and two Ivy Bridge based Intel Xeon cores was used to perform the experiment.

As Gra2MoL supports fewer SQL statements than DBLModeller, the majority of the schemas produced error messages and empty models. Gra2MoL was only able to successfully extract a model from the *University of Murcia Student Management System* [39], which was a sample schema provided with the tool. In order to obtain results, the schemas were modified to remove unsupported content until they could be processed by Gra2MoL.

The mean model extraction times for the DBLModeller and G2M tools are shown in Table 3.6. Every schema was extracted in less time, on average, with the DBLM tool. The largest improvement was seen for the MySQL OFBiz schema which had a 55% reduction in model extraction time (90 seconds). However, the process used to obtain models with G2M reduced the input schema size. In order to perform a like-for-like comparison the "sec / KLOC" metric is used. This specifies how many seconds were required to process 1000 lines of SQL. Interpreting the results in the context of the schema size, increases the significance of the results. The Oracle Science Warehouse

Table 3.7: The SQL elements, and their corresponding KDM model elements, considered during the completeness experiment.

SQL Element	**KDM Element**
Table	RelationalTable
Column	ItemUnit
Unique Key	UniqueKey
Key	ReferenceKey
Sequence	SeqContent
Relationship	KeyRelation

schema shows an 86% reduction in the time required to process 1000 lines of SQL, as seen in the sec / KLOC column. Furthermore, the MySQL MediaWiki schema shows an 84% reduction. RQ3 can be answered by concluding that a significant reduction in model extraction time is achieved by replacing the M2M transformation.

The KDM models produced during the performance experiment were combined with two new SMM models to analyse completeness and correctness for DBLModeller. The SMM models were obtained from two live systems: Wikipedia and Science Warehouse. Load measurements were obtained from Wikipedia by combining the page request logs [170] and database dumps [171] for a six month period. Calendar months were used as the measurement interval. For Science Warehouse a pre-production environment was used and load was replicated using performance tests, which produced a query log for a 31 hour period. From this, a set of load measurements was produced at hourly intervals. The query log was produced using the Oracle JDBC logging tooling [158] and was approximately 3 GB.

In this experiment, completeness is defined in three categories: full, partial, and none. Full completeness requires the same number of: (1) SQL elements and KDM model elements, and (2) workload measurements and SMM model elements. SQL elements are fundamental components of a schema, such as tables and columns. Table 3.7 defines these elements and the KDM elements they map to. Partial completeness is when there are fewer SMM/KDM model elements than SQL elements/workload measurements. The 'none' category indicates that: (1) a model was not created, or (2) it did not conform to the KDM or SMM. Conformance to these metamodels was measured with the Eclipse Modelling Framework.

The experiment also considers the correctness, i.e, whether the data contained in the input text is identical to the data contained in the model. Any corruption or missing data would severely restrict the usefulness of the tool. Each model is therefore judged

```
====================== TABLES ======================
SQL CREATE TABLE:      KDM RelationalTable Elements:
OTG_ACTIVIDADES        OTG_ACTIVIDADES
OTG_MODALIDADES        OTG_MODALIDADES
OTG_PCONVOCATORIAS     OTG_PCONVOCATORIAS
OTG_TCONVOCATORIAS     OTG_TCONVOCATORIAS
OTG_SOLICITUDES        OTG_SOLICITUDES
OTG_PTCONVOCATORIAS    OTG_PTCONVOCATORIAS
OTG_CONVOCATORIAS      OTG_CONVOCATORIAS
OTG_PUBLICAOFI         OTG_PUBLICAOFI
OTG_TERCEROS           OTG_TERCEROS
====================================================
Tables Match? true
```

Figure 3.4: A section of output from ModelChecker.java

as correct or incorrect. This involves comparing the name field for every element in Table 3.7, and any discrepancies mean the model is considered incorrect. Secondly, the values of the workload measurements in the SMM model were compared against those in the input text.

A small model checking tool was created to automate the completeness and correctness analysis (ModelChecker.java in the DBLModeller source code). As input this uses an SQL or CSV file and a KDM or SMM model, and produces a report. For a Structure/KDM model the report lists: table names, column names, primary key and index target columns, relationship target columns, sequence names, and comment values. The report also indicates whether each of these match in the KDM and SQL files. For a Workload/SMM model the Observation elements' name and the DirectMeasurement elements' value are compared against the load measurements in the CSV file. An example of the output report is shown in Figure 3.4 for the University of Marcia schema, and the complete output is available on the project website.

The output from the tool provided the following results:

- The input text and the output model had the same number of elements
- All table, column, and sequence names were correct;
- IndexElements and UniqueKeys referenced the correct columns;
- KeyRelation elements referenced the correct unique and foreign keys;
- Comments (from the COMMENT ON statement) contained the correct text;
- Observation names and Measurement values are correct.

In response to RQ4, it is concluded that the eight models analysed (6 KDM and 2 SMM) are *fully* complete and *correct*. This uses the definitions of completeness and

correctness introduced earlier in this section. Furthermore both models conform to their respective metamodels.

3.3.4 Threats to Validity

Construct validity threats may be due to the assumptions made when evaluating DBLModeller. These assumptions include: (1) that the lines of code in the SQL extension case study require approximately equal implementation effort, (2) that workload data will always be obtainable, and (3) the time complexity of Gra2MoL is $\mathcal{O}(n^2)$. To mitigate assumption 1, the same technologies have been used in both cases (ANTLR and the G2M DSL) and the work undertaken is explained in detail. Furthermore, the functionality of the code is the same in both cases — extraction of a KDM model from a Microsoft Sharepoint schema. To mitigate assumption 2, multiple sources of workload information are identified (log files, monitoring data, and database query logs). All the databases and database infrastructure identified in the literature review (Chapter 2) support atleast one. Assumption 3 is mitigated empirically by extracting models with Gra2MoL from a range of databases. The results in Table 3.6 do not indicate that Gra2MoL has a lower time complexity than DBLModeller. However, a detailed analysis of the Gra2MoL source code would be necessary to establish this conclusively.

Internal validity threats can originate from how the experiments were performed, and from bias in the interpretation of the results. To mitigate these threats, the performance results were repeated over 20 independent runs, and the DBLModeller code and all experimental data and results was made publicly available from the project website in order to enable replication.

External validity is concerned with the generality of DBLModeller. In Sections 3.3.1 and 3.3.3, one concern is whether the sample sets of schemas used are representative of the real world. This has been mitigated by choosing multiple schemas from systems of varying sizes and from different domains. A second concern is that a single case study has been used in Section 3.3.2 to evaluate the extensibility of DBLModeller. To partly address this concern, the case study used a Microsoft SharePoint schema with characteristics common to a wide range of systems. Nevertheless, additional evaluation is required to confirm generality for databases with characteristics that differ from those in the evaluation.

3.4 Chapter Summary

Given the benefits of cloud computing platforms [172], many approaches have been developed for transforming and migrating enterprise systems to the cloud. The most notable are the model-based approaches were developed by the REMICS [25] and ARTIST [26] EU FP7 projects. Although they consider various technical and business issues, support for data and database migration is lacking. Another interesting approach in this area is CloudMIG [21] which can automatically determine a systems suitability for a cloud platform. REMICS, ARTIST, and CloudMIG all make use of the Knowledge Discovery Metamodel, with CloudMIG also using the Software Metrics Metamodel.

KDM and SMM have both been used for other software modernisation purposes besides cloud migration, such as extracting business process mining [173] and extracting requirements [174]. The range of existing tools and approaches using these metamodels make them an ideal choice for this purpose. KDM models can also represent all aspects of a software system (unlike GER [175]), therefore they can act as a common repository for information exchange.

Related work on database knowledge discovery includes: SQL2XMI [176], minimal schema extraction [117], program comprehension from SQL traces [177], and the business knowledge discovery framework from Normantas and Vasilecas [178]. All of these approaches look at discovering lost business and design knowledge from database. There is also work which looks at transforming the database layer, such as Pérez-Castillo et al. who proposed wrapping the database with web services to allow better alignment with service oriented architecture. These approaches do not address the problem of heterogeneity, i.e. there are multiple SQL dialects (e.g. Oracle, DB2) which deviate from the official standard [179].

The two key existing database reverse engineering approaches capable of extracting KDM-based models from SQL, are [39] and [178]. In [39] a grammar-to-model mapping approach is followed so that existing SQL grammars can be reused, while [178] uses an Xtext based solution. Both use a text-to-model transformation, an interminable model, and a model-to-model transformation to produce KDM models. The disadvantage of this two transformations must be developed and maintained; DBLModeller addresses this by using a single annotated text-to-model transformation. Furthermore, to use these tools on the majority of systems they must be extended, which is an expensive task. Engineers must spend a substantial amount of effort on obtaining models instead of using them to modernise the system.

DBLModeller, a tool-supported approach to extracting KDM structure models and

SMM workload models from the database layer of software systems, was presented. KDM [35] and SMM [2] are standardised metamodels developed by the Architecture Driven Modernisation Task Force within the OMG [41], and have been used for a variety of software modernisation tasks. Although most major cloud modernisation and migration methodologies advocate the use of KDM and SMM, their adoption has so far been confined to the code layer of software systems. Furthermore, the extraction of these KDM and SMM models is typically tightly coupled with their use (e.g. SMM is used within CloudMIG but the user is not able to access the model and other SMM models can not be used as input). DBLModeller addresses these shortcomings and accurately extracts models representing the workload (SMM) and structure (KDM) of the database layer of existing systems.

DBLModeller was evaluated using database schemas and log files from multiple real systems. The experiments showed that DBLModeller can extract models from a wider range of systems and can be extended with less effort than the leading existing tool (Gra2MoL). These key benefits were achieved by removing a model-to-model transformation from the model extraction process and by using a single 'dialect-independent' grammar instead of using a grammar for each dialect, respectively. It was noted that other grammars for proprietary SQL dialects are often unavailable, and even if they were available a time consuming and error prone rewrite of the T2M transformation rules would be needed.

Chapter 4

Simulating Cloud Database Migration and Deployment

4.1 Introduction

This chapter presents a discrete-event simulation approach for estimating migration cost, migration duration, and future cloud running costs. The approach, called MigSim, focuses exclusively on the migration and deployment of the database layer, It considers: (1) data transfer into and and out of the cloud, (2) data storage on the cloud, and (3) compute instance/virtual machine usage. MigSim can be equally applied to migrations between clouds, or from on-premise databases to the cloud.

A running example is used throughout the chapter to provide context for the research problems being addressed and to explain the MigSim approach. This example is a fictional CRM application called *CustomerWorld*, which is currently running on physical on-premise servers. The organisation which owns and uses this CustomerWorld deployment wants to migrate it to a public cloud platform. They are currently investigating all the available approaches to perform this migration. The application is written in Java and uses an Oracle database for persistence. The current deployment consists of two application servers and a single database server. This scenario is comparable to the real-world migrations identified in literature review Section 2.3.4.

While planning the cloud migration of CustomerWorld, the time required to move its data from the on-premise Oracle database to a cloud Oracle database must be determined. This depends on the size of the data, the available bandwidth, and the performance of the existing database. Obtaining an accurate time estimate is therefore challenging, but necessary to choose between Internet-based data migration and device

(e.g., HDD) shipping. One common trade-off an organisation might want to investigate is duration versus cost. Additional bandwidth or increased database performance could speed-up data transfer into the new database. Similarly they can look at the cost benefits of 'cleaning-up' the database before migration, i.e., identifying and removing unneeded tables or archiving old data. Accurate costs enable organisations to plan, budget and investigate such trade-offs.

A second challenge is choosing the new cloud provider. There are many cloud cost calculators (e.g., [31, 32]) which can be used to make comparisons based on cost. However, these rely on a user to correctly determine their own resource requirements and input this data. Returning to the CustomerWorld example, while the hardware specification of the database server is known, it is challenging to translate this to cloud instances. The database server hardware is a few generations behind, so its 2.5GHz CPU will have different performance to a cloud instance with the same CPU clock speed. Furthermore, daily or weekly database load trends are unknown. If current growth trends continue at CustomerWorld's parent organisation, it will soon be necessary to move from a single database server to a cluster. An organisation will typically want to consider all these issues when selecting the new cloud provider.

MigSim uses simulation to estimate the migration duration, migration cost, and future cloud running costs. Simulation is necessary because of the large number of variables and their relationships. These include: the source database workload, workload growth, database size, cloud instance performance, and cloud usage charges. In particular the database workload and cloud instance performance are stochastic variables, hence the use of a model transformation/query to obtain the results was ruled-out. Similarly discrete-event simulation was chosen over other types (e.g., Monte Carlo) because the workload is based on time.

From the numerous discrete-event simulators available, the CloudSim Framework [40] was chosen as the most suitable base for this work. It has existing functionality for simulating data migration between software applications running on different hosts. Furthermore, it has been used extensively to simulate other aspects of cloud systems (e.g., [135] and [54]). However, the framework still had significant limitations and extension was necessary.

Functionality has been added to CloudSim to associate costs with resource consumption. These costs are defined in a cloud cost model for the target cloud platform, which is also input into the simulation. The migration duration is dependant on the infrastructure capacity/performance; this performance data is provided to the simulation as a set of parameters.

4.2 Design

4.2.1 Overview

MigSim produces three estimates: (1) the time required to transfer all the data into the new target database, (2) the cost of data transfer and the resources facilitating it, and (3) the running costs for the new target database. Estimate two (migration cost) is the up-front cost of moving the database to the cloud, and includes various charges made by the new cloud platform. These can include VPN devices and middleware servers orchestrating the migration, like the Amazon Database Migration Service [180]. MigSim considers the new target database instance(s) as a migration cost until the data transfer is complete. From this point onwards, it is considered part of estimate three (long-term running costs). The focus of MigSim is on costs which are influenced by the databases' workload. While the cost of training or hiring employees to perform the migration can be significant, several general project management approaches already exist to estimate this [181, 182].

The migration environment within MigSim contains five conceptual components: the software layer, source database, gateway, middleware, and target. These are seen in Figure 4.1 as the: NetDatacenterBroker, InHouseDB, MiddlewareApp, GatewayApp, and CloudDB classes respectively. The Software Layer applies load to the Source Database according to the workload model. The Middleware extracts data from the Source Database by connecting through the Gateway, then inserts the data into the Target Database. The Gateway component corresponds to a server or network appliance that creates a secure VPN tunnel between the source and the target infrastructure (e.g., Amazon Virtual Private Gateway). Data transfer takes place over a simulated TCP/IP network to represent a migration over the Internet. The Source and Target databases have identical schemas.

MigSim has been implemented by extending the CloudSim framework [138]. Any of the leading simulation tools identified in the literature review Section 2.5 could have been used, however CloudSim had some clear advantages. It already supports detailed modelling and simulation of data transfer over a TCP/IP network. Furthermore, many of its existing objects could be re-used to model MigSim's five conceptual components. This is shown in Figure 4.1 where the NetDataCenterBroker is extended to act as the *software layer* component. Many other CloudSim objects are used as-is, including NetworkVM, Host, and Datacenter. Alternative simulation tools did not provide the same level of re-use.

The extensions made to CloudSim include: (1) population of CloudSim objects

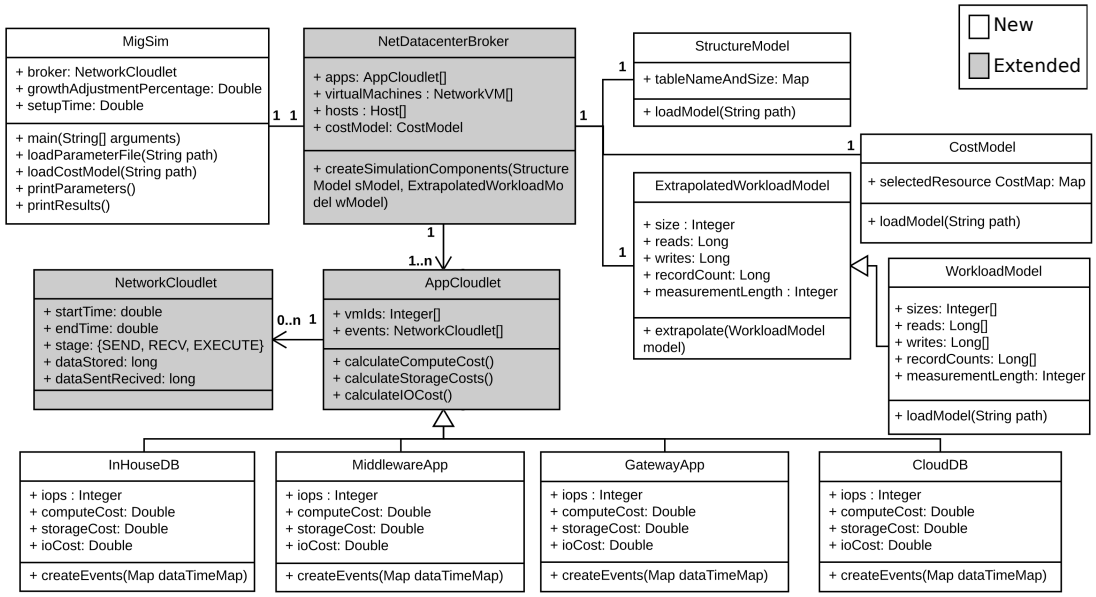

Figure 4.1: New and extended MigSim classes relative to CloudSim (UML class diagram)

from KDM and SMM models, (2) the association of costs with resource consumption, (3) the association of throughput capacity with AppCloudlet objects, and (4) cost calculation post-simulation using data from a cloud cost model. KDM-based structure models and SMM-based workload models now determine the load on the InHouseDB, MiddlewareApp, GatewayApp, and CloudDB. These extensions are discussed in detail in the following sections.

The five conceptual components were included in the MigSim design because they model aspects that have a significant impact on migration costs. The Software Layer affects the available capacity of the source database as it applies load. The source and target databases have a maximum throughput which depends on the underlying hardware, this determines how quickly data can be exported and imported. Furthermore, the Middleware and gateway components become a bottleneck if their performance is worse than the databases. This slows data migration.

Figure 4.2 returns to the CustomerWorld example, and shows two migration options which could be simulated with MigSim. *Option A* is a migration to a database running on the Amazon Relational Database Service. As the data in the CustomerWorld database contains sensitive personal information on customers, the data is transferred to the cloud through a software VPN. The Amazon Database Migration Service (DMS) is used to perform the migration.

Option B is a migration to an Oracle database installed on a Microsoft Azure Virtual Machine. Here the organisation is using Oracle Data Pump [183], which is a data import/export tool included in Oracle databases. Furthermore, data files are being transferred directly between each database using the common SCP protocol. This means the Gateway and Middleware components are not needed. All the migration components in Figure 4.2 are labelled with their corresponding MigSim component.

There is a large variety of databases, system types, and database infrastructure. As mentioned previously, CustomerWorld is a fictional CRM application being migrated from on-premise servers to a public cloud. This illustrates a single use-case, although MigSim aims to provide a generic solution. To this end, MigSim uses a cloud cost metamodel to provide a generic definition of a public cloud's charges. The source database is described using standardised structure and workload models obtained with DBLModeller (Chapter 3).

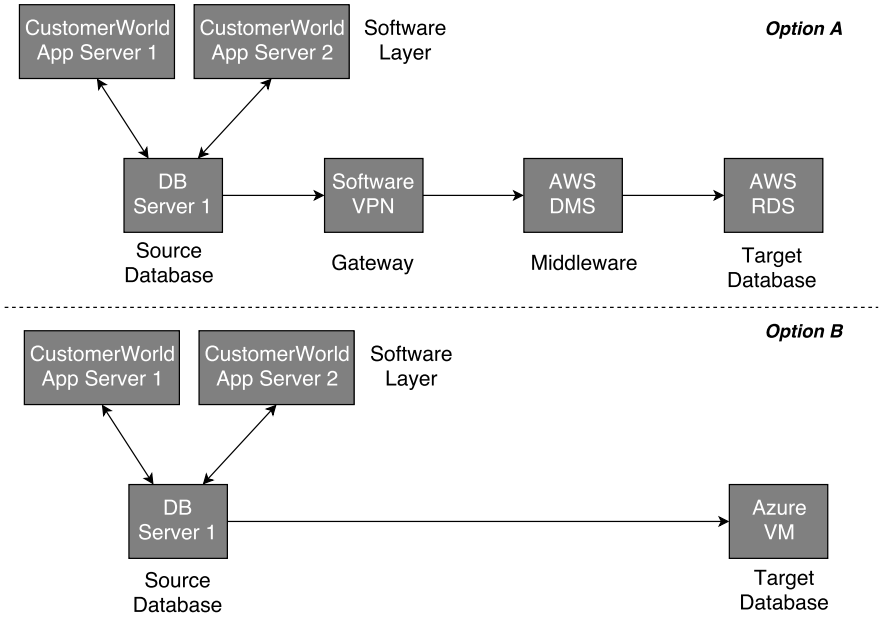

Figure 4.2: Migration components in the CustomerWorld example for two candidate cloud migrations

MigSim has been implemented by using and extending the CloudSim framework. Each migration component from Figure 4.1 is an instance of CloudSim's AppCloudlet object, which can be used to represent an application running in the cloud. Each AppCloudlet is assigned to a single Virtual Machine, running on a separate physical Host, in a CloudSim Datacenter. This allows the use CloudSim's functionality of

simulating data transfer, although this was extended to associate costs with resource consumption. In this implementation components can be removed from the simulation (e.g., the Gateway and Middleware from option B in Figure 4.2). This is performed by configuring the component to have a pseudo-infinite performance and a zero cost.

4.2.2 Workload Simulation

The Software Layer, Source Database, and Target Database components use data from the Workload Model, which specifies historical growth trends and usage patterns. This data must be extrapolated to predict the future database traffic if these trends continue. The storage space consumed by the database in the future is dependant on the amount of new data inserted. Furthermore, the provisioned database throughput required depends on future database traffic. Linear regression is used to estimate the number of future read and write queries received by the database—specifically the ordinary least squares (OLS) method [184].

The OLS method produces a regression equation for the existing data, which minimises the sum of the squared errors (i.e., the error between each data point in the model and the equation). By default the number of estimated future workload measurements is equal to the number of data points in the model. For example, a workload model covering the previous six months with one measurement per month, will be used to produce estimates for the next six months. The state of the Source Database migration component is the same as it was at the last measurement in the workload model. The Software Layer component applies load to the Source Database, which matches workload values estimated with the regression equation. As the simulation runs, the Middleware component migrates the data from the Source Database to the Target Database using slack capacity.

As an alternative to extrapolation from the model, a user can specify an expected growth rate by setting a simulation parameter. This allows business knowledge and plans to be included in the simulation. For example, if CustomerWorld's parent organisation was in the process of acquiring a competitor there would be an increase in the user base. This known and significant increase in load should be considered when planning the cloud migration. They might choose to migrate from a single database to a database cluster immediately. Furthermore, they might select a cloud platform better suited to large-scale databases. The ability to manually specify growth trends means the the MigSim approach captures two sources of knowledge — extrapolated workload models and business plans.

The capacity of the source database, middleware, and target database, is defined in IOPS and input into the simulation. IOPS (or Input/output operations per second) is a performance measurement of storage devices like SSDs and cloud storage services. The throughput of a relational database depends on the performance of the persistent storage, CPU, and available RAM. However, MigSim focuses on the persistent storage IOPS and assumes that other database server components are not a bottleneck. Typically in large enterprise-scale cloud databases the storage costs are greater than those associated with CPU or RAM. The required peak IOPS for an existing database can be measured using the approach in [185].

4.2.3 Parameters

The parameters required for the simulation are listed in Table 4.1; an asterisk indicates two or more values/measurements are required to identify trends. In the MigSim implementation created during this research, these parameters are obtained from the KDM-based structure model, SMM-based workload model, cloud cost model, and command line arguments. However, alternative implementations could use different methods of input while maintaining the design in this chapter.

The 'Required Storage' and 'Database Tables' parameters are used together in the Software Layer simulation component for estimating the size of a typical query. For each measurement the required Storage (in bytes) is divided by he total number of database rows (derived from 'Database Tables'). The Software Layer also extrapolates the future quantity of read and write database queries by using historic data from their respective parameter sets. The result of these two calculations determines the size of each query in the simulation and the volume of queries sent between the Software Layer and Database Layer.

4.2.4 Results

Once the simulation is complete, the costs are calculated from the migration duration and the utilised cloud resources. MigSim only calculates the costs associated with the new infrastructure, i.e., the migration middleware, the VPN gateway on the new cloud platform, and the new database. The migration will increase the load on the existing database and utilise its Internet bandwidth, although the cost of this is heavily dependant on the organisation so it is not considered within the simulation method.

Additional compute time is included in final predicted costs for set-up, configuration, tear-down. As part of these activities the infrastructure may need to be kept

Table 4.1: Simulation parameters and their description.

Paramater	Description
Required Storage *	The storage capacity required by the database (e.g., in GB).
Read Queries *	The number of read queries within a measurement period.
Write Queries *	The number of write queries within a measurement period.
Database Tables *	The name of each database table with the number of rows it contains at measurement point
Costs	The charges of each available cloud resource on the target cloud platform.
Cloud Resources	The cloud resources to use for: the target database, VPN gateway, and middleware. These are subset of the cloud resources in 'Costs'.
IOPS	The IOPS for the chosen cloud resources.
Additional Time	The additional time the migration infrastructure is required to running before and after the migration (e..g, for setup or overnight).
Growth Rate	The percentage that the volume of read write queries will increase by. By default this is derived from *Read Queries*, *Write Queries*, and *Database tables*. It can also be specified manually.

running during non-working hours (e.g., overnight, weekends, or holidays). The *Additional compute time* parameter is manually specified by the user. It should be set according to the experience of the team; if they have previously worked with the system and cloud platform less time may be required. These activities are a necessary part of any migration and can significantly increase the total migration cost for small databases.

4.3 Cloud Cost Metamodel

A metamodel has been developed to define the structure and content of the cloud cost model. The wide ranges of services typically offered by cloud providers and the regional variations in pricing is complex; this makes a model-based approach for including cost data in the simulation desirable.

The metamodel — implemented in the Eclipse Modelling Framework — has been designed to support charges from major public clouds which may be incurred during

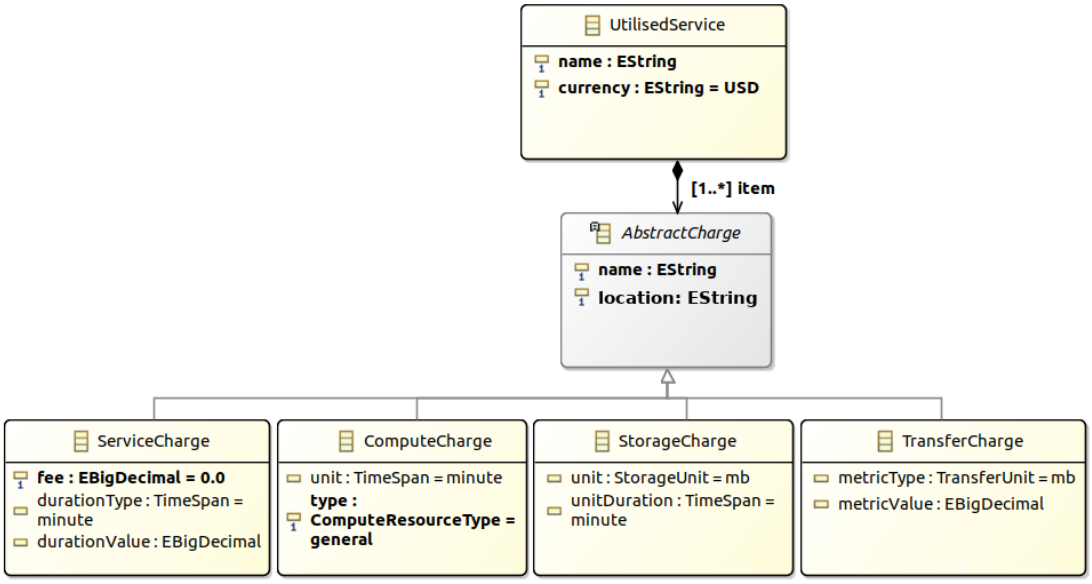

Figure 4.3: A snippet of the cloud cost metamodel showing its key components

a database migration. Each instance of the metamodel corresponds to a single cloud provider, and covers a set of cloud services (e.g., Amazon RDS and EC2). Finally, each cloud service is associated with multiple cloud charges, as shown in Figure 4.3.

Negotiable and market-based cloud charges have been excluded from the metamodel scope. The majority of organisations are unlikely to deploy, and thus to pay for a relational database deployment in this way. For example, Amazon's EC2 spot instances are automatically terminated when the current market price exceeds an organisation's bid price. They are intended for transient workloads or those which can be accelerated when compute instances are cheaply available. Furthermore, negotiated/private pricing arrangements are typically the domain of extremely large-scale systems (i.e., hundreds of millions of users worldwide). These are outside the scope of MigSim due to their scarcity and unique characteristics.

Four types of charge are relevant to database migration and therefore captured by the cloud cost metamodel: service, compute, storage, and transfer. A 'ServiceCharge' represents non-infrastructure charges, e.g, an AWS Support Plan, static IP addresses, or VPN connections. This consists of a fee and a duration which can be specified in minutes, hours, days, months, or years. A 'ComputeCharge' models the cost of a cloud Virtual Machine running for a period of time (e.g., Amazon EC2), while a 'StorageCharge' corresponds to the cost of persistent data storage (e.g., Amazon S3). A 'TransferCharge' represents the cost for data sent or received across a network.

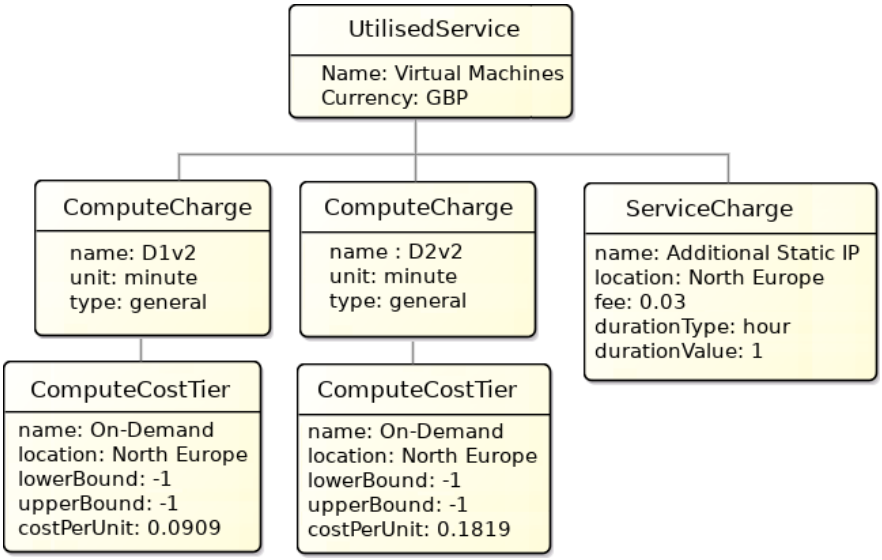

Figure 4.4: A snippet of the cloud cost model for Microsoft Azure

Charges may be tiered so that a per-unit discount is applied for heavy usage or to allow regional price differences.

As part of the MigSim implementation two metamodel instances were created, one for Amazon Web Services and one for Microsoft Azure. These models contain real values taken from the providers' websites, and enabled the simulation migration of multiple databases between the two platforms. A snippet from the Azure cost model is shown in Figure 4.4, here a single Azure service is modelled (Virtual Machines). The 'ComputeCharge' elements represent two virtual machine instance sizes, called D1v2 and D2v2. Each has a single ComputeCostTeir, although a large model might have several per ComputeCharge to accommodate different regions and free-usage tiers. The AWS model is available in Appendix A and the complete Azure model is available in Appendix B.

Several existing metamodels also capture cloud costs, such as those proposed by Leymann et al. [34] and Maximilien et al. [65]. However, these metamodels are not specific to the cloud migration of databases and model elements unnecessary for the use case are present in each metamodel. Rather than using an existing metamodel 'out-of-the-box', one possibility is the modification/tailoring of it to support cloud cost modelling. After investigation this was ruled out because: (1) the effort for approach users would be greater, and (2) the tailored metamodel would be less intuitive. MigSim requires users to populate a cloud cost model from data on a provider's website, there-

fore ease-of-use was considered to be essential.

4.4 Evaluation

4.4.1 Migration Case Study

The key goal of MigSim is to accurately estimate cloud database migration cost and duration. This would enable different migration options and parameters to be evaluated, therefore supporting decision-making. The extent to which this goal has been achieved, was meaured by comparing the MigSim simulations to actual cost and duration values obtained by migrating two systems between two public cloud providers. The first is a closed-source system from Science Warehouse [186] (the industrial partner for this research project) and the second is the open-source ERP System Apache OFBiz [165]. The prediction results have been compared against real figures from these migrations.

Both databases (Science Warehouse and OFBiz) were migrated twice: from the existing cloud platform to a new cloud platform, then back again. This provided two data points per system for the evaluation. While this evaluation used public clouds as the source and target infrastructure, it can also be applied to in-house to cloud migrations. The Amazon Database Migration Service was used as middleware to perform both migrations. Additionally, the Science Warehouse migration required a VPN between the two clouds to secure the data during transfer.

The Science Warehouse system is an enterprise procurement system for making purchases in business-to-business scenarios. At the centre of this is a product catalogue which is populated by 'supplier' organisations, from which 'buyer' organisations can make purchases. The vast majority of users are in the United Kingdom and Ireland, resulting in peak loads during business hours in these countries. The Oracle database from the company's development environment was modelled using the DBLModeller approach and migration was simulated with MigSim. It was migrated for real as part of this research project.

Apache OFBiz (Open For Business) is an ERP system which contains applications for: e-commerce/online shopping, fulfilment of orders, marketing, and warehouse management. Unlike with Science Warehouse a real instance was unavailable; therefore the system was populated with 200GB of synthetic data to represent the use-case of large online retailer. This was random data which matched the purpose of the column, e.g., 16-digit numbers following the Mastercard and Visa format were inserted into a credit card number column. All database tables were populated.

A snippet from the Apache OFBiz workload model is shown in Figure 4.5 to illustrate its structure. The complete model (published at [9]) contains 12 observations at monthly intervals. Each observation has three 'Observed Measures' which contain at least one measurement. Both the OFBiz and Science Warehouse models were produced using DBLModeller (Chapter 3) and conform to the Structured Metrics Metamodel.

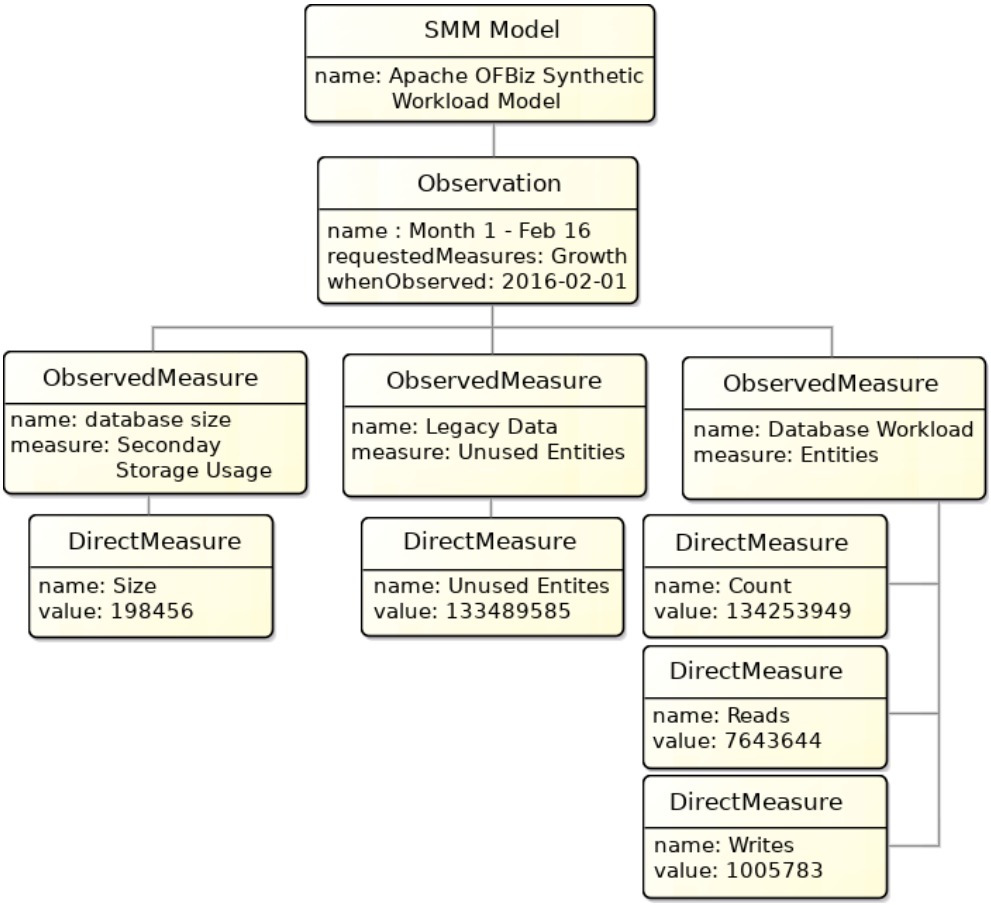

Figure 4.5: A snippet of the Apache OFBiz Workload Model

The Science Warehouse database was migrated while idle and the OFBiz database was migrated with synthetic load applied. This reflects how some of the approach users can perform the migration with the system shutdown. However, many larger systems would be critical to the organisation and this would be impossible. The load represented a user base within a single country, with two daily peaks at approximately 1100 and 1500. A daily total of 1.3 million queries were made; 90% between 0600 and 2000. Two Amazon EC2 instances were used to send these queries to the database

server.

The four migrations are modest in terms of size and cost. Many organisations migrating enterprise system databases will be moving data between database clusters rather than single servers, making the cost more significant. A common reason for database migration is scalability, where database load is reaching capacity. Limited capacity would therefore be available to migrate the database while it is being used. Inducing such large databases and loads was not feasible in this evaluation due to the high costs. However, it is expected that the accuracy of the experimental results would be similar for larger systems (such as those migrated in [187] and [122], which have similar characteristics).

Table 4.2: Simulated Migration Costs

System	Migration	Size	Migration		Additional Time		Total Cost
			Mean Duration (Std Dev.)	Cost	Hours	Cost	
Science Warehouse	AWS → Azure	38GB	417 Min (7 Min.)	$4.70	16	$28.26	$32.96
	AWS ← Azure	18GB	144 Min. (2 Min.)	$3.77	16	$22.64	$26.41
Apache OFBiz	AWS ← Azure	200GB	1147 Min. (12 Min.)	$11.43	1	$0.67	$12.10
	AWS → Azure	163GB	901 Min. (8 Min.)	$9.08	1	$0.60	$9.68

Table 4.2 presents the estimated cost and duration for each migration, obtained

Table 4.3: Actual Migration Costs

System	Migration	Size	Migration Duration	Additional Time	Total Cost
Science Warehouse	AWS → Azure	38GB	402 Min.	12.8 Hours	$40.12
	AWS ← Azure	18GB	147 Min.	16.15 Hours	$30.75
Apache OFBiz	AWS ← Azure	200GB	1176 Min.	1.1 Hours	$13.30
	AWS → Azure	163GB	888 Min.	0.7 Hours	$6.96

Table 4.4: Anticipated set-up tasks for the migrations

Migration	Anticipated Tasks
Science Warehouse: AWS → Azure	Create Azure VM. Install Oracle. Configure VPN. Test connectivity.
Science Warehouse: AWS ← Azure	Create RDS DB. Test connectivity. Leave idle for non-working hours.
Apache OFBiz: AWS ← Azure	Create RDS DB. Test connectivity.
Apache OFBiz: AWS → Azure	Create Azure VM. Install Oracle. Test connectivity.

with MigSim. Each simulation was performed 20 times on a laptop PC with a Intel i5-6200 (dual core, 2.3GHz) and 8GB of RAM. Execution times ranged from 8 minutes for the 18GB simulation (best case) to 144 minutes for the 200GB simulation (worst case). When computing the cost, MigSim rounds the migration duration up to the nearest full hour, in line with the cost model of many cloud providers. Therefore, the cost did not vary between runs, although small differences may arise for larger datasets.

The *Additional Time* parameter was manually estimated for each migration prior to its simulation, as proposed in Section 4.2. The migrations were performed by a researcher with some knowledge, but no working experience, of the Amazon Database Migration Service. However, the researcher had experience using AWS RDS, EC2, and Azure VMs. Science Warehouse allocated a fixed time-box for the experimental migration of their database (i.e., one weekend). Their database had to be secured using a VPN and the databases had to be inside a virtual private cloud. No time-box or security requirements existed for the OFBiz migration as it only contained synthetic data. Based on this information, a list of anticipated tasks (shown in Table 4.4) was constructed, and the sum of their anticipated durations was used as the *Additional Time* parameter of MigSim.

A side-effect of database migration is that the target database will consume less storage space and perform better than the source database (despite identical data). This is due to different amounts of data being inserted and removed during routine usage of the source database, creating fragmented free space [188]. Furthermore, all schema objects are (tables, indexes, etc.) are essentially rebuilt in the target database. The size column was included to show the storage space consumed by the database before it was migrated. This value was provided as input to the simulation via each system's workload model.

Table 4.3 presents the actual cost and duration for each migration. These values

were obtained using data from Amazon CloudWatch and the Microsoft Azure Active Log, which are services that record the creation and deletion times of each cloud resource. The Science Warehouse outbound migration took 0.3 hours less than predicted and its "return" migration took 0.15 hours less; a relative error of of 4% and 22%, respectively. In contrast, the outbound OFBiz migration took 1.6 hours longer then predicted and the return migration took 0.8 hours longer; these figures correspond to a small relative error of 8% and 5%, respectively.

Performance Impact

The OFBiz database was much larger and subject to load, compared to the Science Warehouse database. However, the increases in migration duration and cost (presented in Table 4.2 and Table 4.3, respectively) were modest. These results are heavily influenced by database performance. The total cost results are influenced by additional time in the experiments, as discussed previously.

In the evaluation it was necessary to use different cloud database instances types as: (1) the source Science Warehouse database was controlled by the business, and (2) the databases were different sizes. The performance of a cloud storage device is typically proportional to the allocated capacity [189] [190]. The Science Warehouse database used previous generation HDD-based storage, which provided significantly reduced performance compared to solid state drives (SSDs).

One solution would be using the Science Warehouse database instance type across every migration and system. However, a HDD-based device would be unrepresentative of modern cloud-to-cloud migrations. Instead, it was decided to match the database type to the database storage requirements.

Table 4.5 shows: the storage devices used during the database migrations, their published performance (in IOPS), their cost, and a performance factor. The costs were obtained at the time of migration for AWS and Azure datacenters located in Ireland. The performance factor is based on the best performing disk used in the experiments. It represents how much longer a database instance would need to run to export the same quantity of data. For example, an Amazon RDS database using magnetic storage requires 23 times longer to read the same volume of data as the fastest Azure disk.

Table 4.5: Storage devices used during the evaluation with their performance and cost

Storage Device	IOPS	GB-Hour Cost (USD)	Factor
AWS Magnetic Storage	100	$0.0002	23
AWS Provisioned IOPS SSD	1000	$0.0017	2.3
Azure P6 Premium Managed Disk	240	$0.0001	9.6
Azure P20 Premium Managed Disk	2300	$0.0002	1

Typically IaaS cloud providers offer several generations of storage and compute products, which reflects the different generations of hardware in their datacenters. The latest generation hardware provides the best value (i.e., the highest performance:cost ratio). This is seen in Table 4.5 where Amazon's newer SSD-based storage delivers ten-fold increase in I/O operations for only 8.5 times the cost.

The evaluation compares like-for-like hardware so the performance differences do not affect the results. For example, the AWS to Azure migration of the Science Warehouse database uses magnetic storage (Table 4.3). The simulation of this migration models the performance of magnetic storage (Table 4.2).

4.4.2 Running Costs

In this section the accuracy of the estimated cloud running costs produced by the MigSim tool-supported approach are evaluated. Running a system on multiple clouds for a long period of time (with additional servers applying a synthetic load) was not feasible for this evaluation, so instead the focus is on the auto-scaling accuracy. The simulation considers the future database load (extrapolated from the workload model) and the capacity of the new database server. Once the load exceeds capacity an additional server is introduced i.e., auto-scaling takes place. The error between simulated auto-scaling and actual auto-scaling is the main factor influencing the accuracy of the running costs. For example, if the simulation underestimated the database performance then servers would be added too soon and inflate the predicted costs.

Table 4.6 presents the predicted running costs obtained for Apache OFBiz on Amazon Web Services and Microsoft Azure. These values are based on the synthetic OFBiz models that were created using the DBLModeller method before cloud migration. Figure 4.6 shows how these costs are incurred over time.

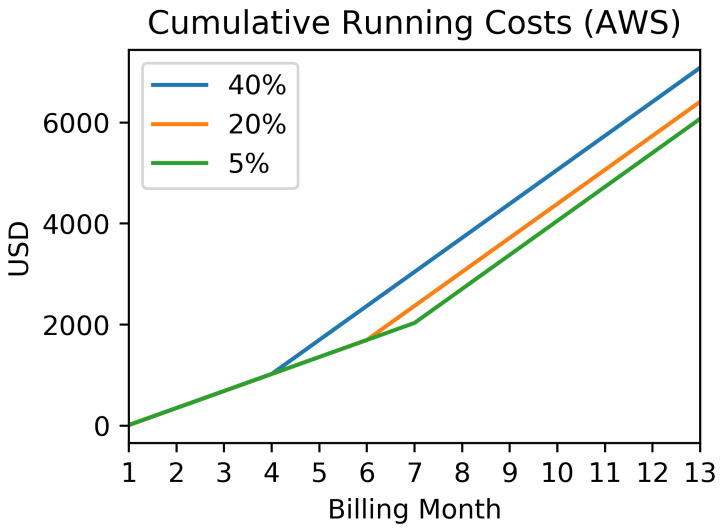

Figure 4.6: Cumulative cost for an AWS deployment of Apache OFBiz

Table 4.6: Predicted cloud running costs for Apache OFBiz (inc. migration)

Platform	Growth	Mean Cost (Y1)	Mean Scaling Point	Scaling Point Std. Dev
AWS	40%	$7,084	Week 16	0.46
	20%	$6,410	Week 24	0.39
	5%	$6,073	Week 28	0.46
Azure	40%	$15,003	Week 16	0.39
	20%	$13,574	Week 24	0.46
	5%	$12,860	Week 28	0.46

By default MigSim produces cost estimates based on the assumption that the observed load trends in the model will continue. The growth rate can also be manually adjusted to accommodate business plans. The results in Table 4.6 use this functionality to produce the 5% and 40% growth rates. These represent potential business scenarios where growth is higher or lower than expected. The results for 20% growth represent the actual trend in the model.

In order to assess the accuracy of these results, a single OFBiz instance was setup and the load from Table 4.6 was applied for the 20% growth rate. The queue length and error logs for the Oracle database were monitored. At this load the database should be below capacity, therefore the queue should be stable and no timeout errors should

be thrown (ORA-12170); these characteristics were confirmed in the experiment. The load was then ramped up by 1 query/sec every five minutes, until a timeout error was thrown by the application.

The target database performance was set to 100 IOPS, which matched the storage performance of a HDD-backed AWS RDS instance [189] and a HDD-backed 'standard disk' in Azure [190]. The storage devices with the lowest level of performance were selected to reduce the evaluation costs—these devices required the least synthetic/generated load to reach full capacity, and therefore fewer cloud instances had to be paid for.

In MigSim, 100 IOPS equates to an auto-scaling threshold of 24 queries per second for each database server. This threshold remains constant in every simulation, although it is reached at time moments that depend on the growth rate (sooner for high growing rate, as shown in Figure 4.6). In this definition, a query is a typical CRUD (create, read, update, or delete) SQL query for an ERP system. Batch, back-up, and replication tasks are excluded. The calibration between published database performance (IOPS) and typical database queries per second is a fixed-ratio. The default value for this ratio was determined during the design phase of MigSim by analysing the database traffic from Science Warehouse and Apache OFBiz.

On Amazon AWS, the first timeout error was logged at a load of 27 queries per second (cf. Figure 4.7). Database capacity was therefore reached 3 queries/sec after it was in the simulation. This is a relative error of 12% and meant that the simulation "launched" a new instance six weeks earlier than necessary. This would result in the simulated cost likely being $246.96 higher than actual running costs. However, scaling-out a database from 1 to 2 instances is a time consuming and non-trivial task. It is expected that this operation would take multiple weeks, therefore reducing the error.

On Microsoft Azure the first timeout error occurred when the load reached 26 queries per second, i.e., a relative error of 8% compared to the simulation results. This would result in the predicted cost being $722.24 higher than the actual cost for this scenario.

The accuracy of auto-scaling depends on multiple factors, including the performance of the cloud instance [191], the size and structure of the database, and the queries being made to the database. However, it is expected that an error of up to ±12% to be typical for most enterprise web applications when modelling database performance on published IOPS.

This evaluation accommodates the unfeasibility of performing a long-term cloud migration case study with a real organisation, although it has some limitations. The target

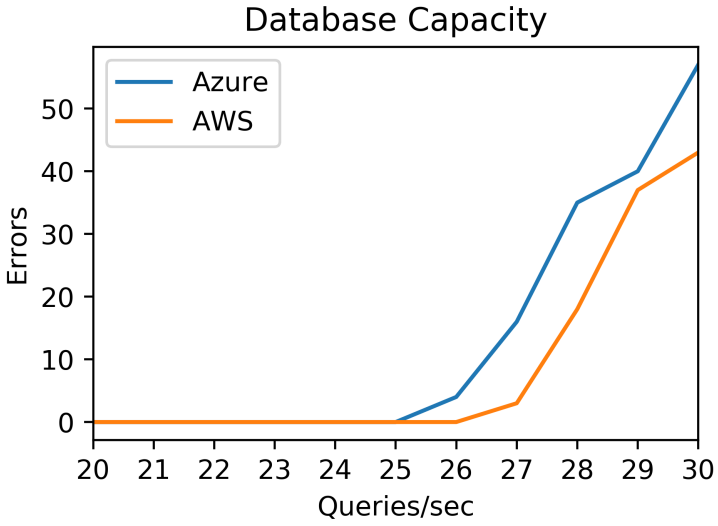

Figure 4.7: Timeout errors received at different load levels

system (Apache OFBiz) has a relatively low workload requiring one or two database servers. Larger systems will likely see scaling up and down on a daily basis. Furthermore, testing auto-scaling required synthetic models. Additional experiments would be needed to further conform the promising results are also seen in larger deployments.

4.4.3 Threats to Validity

Construct validity threats may be due to simplifications and assumptions made when evaluating MigSim. During the evaluation of cloud migration costs and duration,the predictions were compared against real-world migrations. On the other hand, cloud running cost accuracy was inferred from database capacity accuracy (for the reasons presented in the results analysis from the previous section). Real long-term experiments to assess this cost would be necessary to increase the confidence that this result is accurate.

Internal validity threats can originate from how the experiments were performed, and from bias in the interpretation of the results. To mitigate against these threats, the experiments prone to such bias (i.e., the database migration simulations with MigSim) were repeated over 20 independent runs. The code, experimental data, and results have been made publicly available in a GitHub repositories at [10] to enable replication. While the real-world cloud database migrations used to evaluate the estimated migra-

tion cost and duration were small-scale, much larger cloud migrations were referenced when interpreting the experimental results. It should also be noted that migrating databases from ERP systems is an expensive and time consuming process. Full advantage was made of the systems by migrating them twice, although experiments on other systems should performed to confirm the encouraging findings.

External validity is concerned with the generality of the approach. The characteristics of the closed-source system used in this work (Science Warehouse) have been explained in detail so the results can be generalised. The migration and running cost evaluation focused on Oracle databases although since the approach abstracts away from the database implementation it can be applied elsewhere. MigSim considers the database capacity, load, and size; these are all implementation-independent properties. Furthermore, two cloud platforms were used during the evaluation (Amazon Web Services and Microsoft Azure) and it is argued that the approach is compatible with other IaaS and PaaS providers.

4.5 Chapter Summary

Current cloud cost calculators [31] [32] and comparison approaches [192] rely on their users to know which resources they require and when. However, it is rarely possible map the hardware specification of on-premise servers (or cloud instances from a different provider) to new cloud instances. The migration of a database to a new cloud provider will require temporary migration infrastructure which is separate from the new cloud deployment (e.g., a VPN, Amazon Database Migration Service [180] , or Amazon Snowball [22]). No existing approaches can predict the cost of the infrastructure which depends on the length of time it is required. This includes: data transfer duration, configuration, testing, and staff holidays. To contribute towards the solution of this problem, this chapter presented MigSim, a tool supported-method that simulates cloud database migration.

MigSim extended the CloudSim discrete event simulation framework and adds several features necessary for cloud database migration and deployment simulation. These include: (1) obtaining parameters from workload (SMM-based), structure (KDM-based), and cost models, (2) associating costs with resource consumption, and (3) modelling database performance. The simulation design contains four components: the Software Layer, Source Database, Gateway, Middleware, and Target Database. During migration simulation data is sent from the Source Database to the Target Database through the Gateway and Middleware. The Software Layer applies load to the Source

Database according to the workload model, which represents the load the system may be subject to in real conditions.

During deployment simulation the Software Layer applies load to the Target Database component which represents the new cloud database. The result of the simulation is predicted values for: data transfer duration, migration cost, and future cloud running costs. Historic growth trends observed in the workload model are considered for the the future cloud costs, e.g., if there was a 10% growth in database traffic in the last year then it is assumed there will be a 10% growth next year. However, users can adjust this growth rate manually to reflect changing business conditions.

MigSim was evaluated by migrating two enterprise systems between Amazon Web Services and Microsoft Azure, and comparing the migration cost and duration to the MigSim predictions. A database capacity experiment on was performed on Amazon's Relational Database Service and Microsoft's Azure Windows Virtual Machine Service. The estimated migration cost and duration had a relative error between 4% and 22%, while the estimated cloud running costs had a relative error of between 8% and 12%. It was argued that these estimates would be valuable to organisations when planning a database migration from a physical server to the cloud, or between cloud platforms.

Chapter 5

Cloud Database Migration Planning Methodology

5.1 Introduction

The existing approaches for evaluating cloud migration options, such as CloudMIG in Section 2.3.3, do not support the database layer. As a result organisations may perform this task in an ad-hoc manner, with unknown accuracy. Typically the organisation must decide upon several database migration options, such as: data transfer method, downtime tolerance, deployment method, cloud instance size, and cloud platform. Failure to make an accurate decision will lead to over and under provisioning of resources, and therefore increased costs or SLA violations. However, this is a non-trivial task requiring a detailed understanding of the database structure, size, workload, and growth.

This chapter presents a novel methodology for evaluating the impact of migration decisions on the cost and duration. The methodology integrates the approaches and tools from Chapters 3 and 4, providing an end-to-end solution. The focus is on relational database migration over the Internet, where the destination database is running on chargeable cloud resources. The source may be an existing cloud database or a database running on an organisation's internal infrastructure.

The methodology is intended to be used during the planning phase of a software system cloud migration project. Each set of proposed migration options provided as input, results in: the migration duration, migration cost, and future running cost. These can be compared and used to make decisions in the migration project.

This chapter is structured as follows. Section 5.2 outlines each phase and task in

the methodology, as well as the notation used. Section 5.3 discusses how this work has been evaluated and considers threats to validity. Finally, Section 5.4 summarises the key points and conclusions.

5.2 Methodology

The methodology has three phases, each containing multiple tasks which are performed by the software engineers migrating the database. These are shown in Figure 5.1 and explained in the following subsections. The methodology is defined using the Eclipse Process Framework [193], as this notation is used by other migration methodologies (e.g., ARTIST [26] and REMICS [83]). The framework automatically generates documentation (as HTML) and process diagrams.

Figure 5.1: Methodology phases

5.2.1 Planing

The Planning phase has three tasks, as shown in Figure 5.2. As part of the *Select Simulation Goal* task, the methodology user should decide which migration options to evaluate. This can be a single set of migration options, for example, where they wish to obtain duration and cost estimates for an existing migration plan. Alternatively, several sets of migration options can be compared to choose the most suitable.

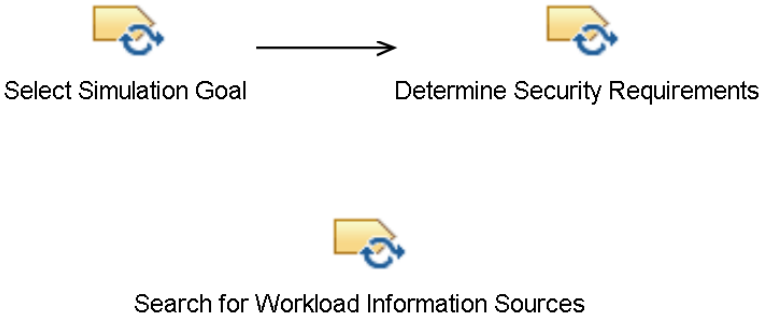

Figure 5.2: Key tasks in the planning phase

The following cloud database migration options should be considered:

- The target cloud platform.
- The regions or datacenters within the target cloud which are suitable for use with this system.
- Cloud database instance type(s), i.e., the hardware specification of the database. This can be estimated.
- Deployment type, i.e., will the database be deployed on a Database-as-a-Service or configured to run on top of a cloud virtual machine.
- Migration infrastructure, i.e., the tool used to migrate the data. This may be provided by the target cloud (e.g., the Amazon Database Migration Service), the database vendor, or a third party. A manual migration where data is copied directly is also possible.
- A downtime or zero-downtime migration.

Future business plans can impact the number of users of a software system, which consequently impacts database load. Estimated database size or workload changes can also be included in the simulation goal. The option and business plan information should be recorded, an example is shown in Table 5.1 where two sets of migration options are compared.

Table 5.1: Option sets for the Science Warehouse database migration in Section 4.4

	Simulation A	*Simulation B*
Target Platform	Microsoft Azure	Amazon Web Services
Database Deployment	Virtual Machine	Database-as-a-Service
Database Instance	D4v2	m4.large
Downtime	Weekend	Whatever is necessary
Infrastructure	Amazon DMS	Amazon DMS

The information captured in Table 5.1 (or in a similar format) should be used to create a simulation goal. Example goals include 'Determine if Amazon RDS or Azure SQL database will be cheapest for our database workload' and 'Estimate the running costs for our planned AWS migration so we can budget for the next tax year'. The goal will guide the later steps in the methodology.

The *Determine Security Requirements* task can only be performed after *Select Simulation Goal*, as the chosen target cloud provider(s) must be known. It requires users

to identify whether their data must be: (1) encrypted during transfer and/or (2) encrypted at-rest. They must also consider how these requirements can be met, which will depend on the source infrastructure, target cloud, and database type. Some systems may not require encryption, such as those aggregating publicly available data. The outcome of this task is a list of the chargeable cloud resources needed to satisfy the security requirements.

In a cloud database migration the implementation of security requirements typically has a cost. This cost is challenging to identify and compare in different cloud providers, hence the need for this explicit step. Cloud providers will have different services with different properties. For example, a hardware VPN might be necessary to meet the security requirements on Cloud A. However, cloud B might not offer this service which forces the organisation to use a cloud VM with a software VPN installed. The methodology does not contain explicit tasks for other types of requirements as they are often organisation and system specific, whereas all organisations will have data to determine the security requirements for. These non-security requirements should be reflected in the migration goals in the previous task.

Search for Workload Information Sources can be performed at any time during the phase. Here the user must review the logging and monitoring data produced by the existing system, and determine if the database workload can be measured or inferred. The data must contain: the quantity of data inserted into, and read from, various database entities within a time box. The database entity could be a table, set of tables, or schema. The time box could be hourly, daily, weekly, or some other regular period. The purpose of this data will be to identify database workload fluctuations and long-term growth trends; as a result there should be data points covering multiple months.

The output of the *Search for Workload Information Sources* task is a list of verified information sources and the length of time they cover. Furthermore, the user will have decided whether the combined group of information sources is sufficient, partly sufficient, or insufficient. This judgement is based on the required data contents mentioned previously.

The planning phase of this methodology was performed for the cloud migrations in Section 4.4.1. This presents example security requirements and workload information sources, as obtained from our partner Science Warehouse. In this case the simulation goal was to determine the most cost-effective cloud platform to host their database. The results would then assist Science Warehouse to select a platform to invest in.

5.2.2 Modelling

The modelling phase extends the DBLModeller approach (described in Section 3.2) and contains four tasks as shown in Figure 5.3. The extensions include: (1) definitions for the output of each task, (2) criteria for using query logging, and (3) additional guidance. DBLModeller is a generic approach for database modelling, hence using it to determine cloud migration and deployment costs requires specialisation.

First, the approach user must perform *Schema Extraction*, *Query Logging* and/or *Obtain Workload data*. If there were sufficient sources of workload information (as decided in the previous phase), then the user does not need to perform the *Query Logging* task. However, if this information was only partially sufficient then the user should perform *Query Logging* and *Obtain Workload data*, then combine the results. The process of combining the results is system-specific and therefore left to the user, although this should produce data in the format described in Table 5.2. The *Obtain Workload data* task does not need to be performed if there were no existing workload information sources identified.

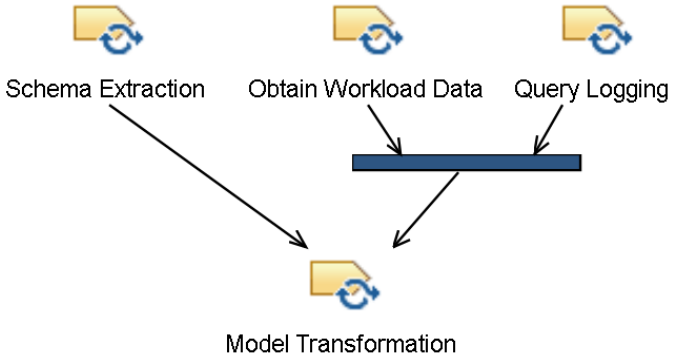

Figure 5.3: Key tasks in the modelling phase

The *Schema Extraction* task requires a user to extract a SQL schema from the existing database using a suitable tool. For example, Oracle SQL Developer [157] can extract a schema from Oracle databases and MySQL Workbench [156] can extract a schema from MySQL databases. Multi-platform tools can also be used [152]. The result should be a single file containing SQL statements to create the database tables, columns, and relationships. Other schema components (e.g., indexes) can be excluded. Existing SQL scripts, such as those in a version control system for recreating the database, should not be used as they may be outdated.

In *Obtain Workload Data* the user must obtain the workload data from the previ-

Table 5.2: The required CSV file format for output of the 'Obtain Workload Data' task.

Line	**Data**
1	System Name (String), Scope (String), Entity Name (String)
2 ... n	Measurement Period Alias (String), Measurement Period Start (Timestamp; yyyy-mm-dd hh:mm:ss), Measurement Period End (Timestamp; yyyy-mm-dd hh:mm:ss), Entity Count (Integer), Entity Reads (Integer), Entity Writes (Integer), Unused Entities (Integer), Database Size (Double)

ously identified sources. This should be processed to produce a CSV/Text file with the format shown in Table 5.2. This task — and the whole modelling phase — has been performed during the DBLModeller evaluation in Section 3.3. Example CSV files are available from [9].

Query Logging is a method for providing detailed workload information by intercepting queries sent from the software system to the database and recording them. Several third-party tools can perform this task, the output should be a list of SQL queries and the time they were made. The length of time this data should be captured depends on its role: (1) to record short-term fluctuations in workload, e.g., daily usage patterns, or (2) to record medium-term workload growth trends. The definition of short-term and medium-term depends on the target system, so the number of system users and their time zone must be considered. Typically the first role requires a few days of query logs, while the second requires a few months.

Once the methodology user has identified a duration for the current system they can begin query logging. The resulting logs must be processed into the Table 5.2 format. This can be performed automatically by the DBLModeller tool for query logs in the P6Spy format [164]. P6Spy was chosen as it had the least performance impact when used with the systems in this research (as discussed in Chapter 3). Other tools could be used but DBLModeller would need to be extended or a new tool developed.

The resulting data (in a CSV file) must satisfy the two goals, i.e., (1) short-term fluctuations and (2) medium-term growth trends. However, this does not need to be done in a single query logging period. A methodology user may choose to have a high-

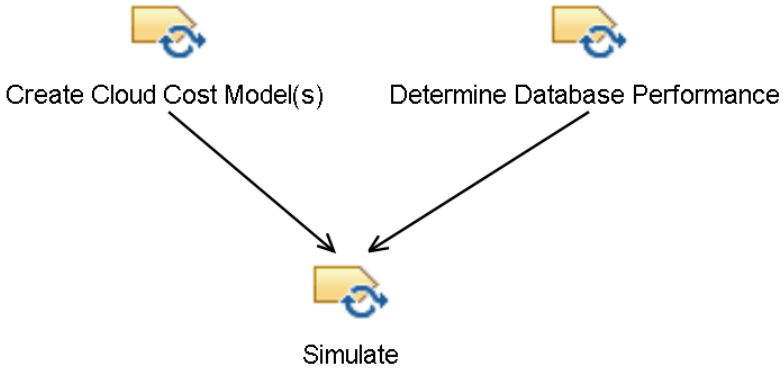

Figure 5.4: Key tasks in the simulation phase

fidelity query logging period to satisfy role 1 and a low-fidelity query logging period satisfying role 2. The fidelity (i.e., detail of query logs) can be adjusted in most tools. Another possibility would be to use query logging for role 1 and another source of workload information for role 2.

The *Model Transformation* task must be performed last as it uses the outputs from the previous tasks (a SQL schema and a CSV workload data file). It has been automated using the DBLModeller tool which performs two Text-to-Model transformations. The result is a database structure model conforming to the Knowledge Discovery Metamodel [1] and a workload model conforming to the Structured Metrics Metamodel [2].

5.2.3 Simulation

Simulation is performed using the MigSim tool (Chapter 4), with the models obtained in the previous phase. The workload model defines the database size and the queries it received during a series of measurement periods. The structure model defines the database tables and their properties. The data in the models determines the properties in the simulation, as described in Section 4.2.

Additional information is required beyond the database models, therefore two pre-requisite tasks exist: create cloud cost model(s) and determine database performance. This is illustrated in Figure 5.2.3.

Create Cloud Cost Model(s) requires a user to produce a cost model (conforming to the metamodel proposed in Chapter 4) for each cloud platform specified in the simulation goal. This model should at least contain the costs and cost tiers for the cloud instances to be used. However, if several instances or regions are to be used then it may be easier to model the complete service (e.g., Amazon RDS) instead of trying

to model the correct subset. The model can be created manually with data obtained by visiting the cloud providers website or from cloud comparison services. Figure 4.4 in Chapter 4 shows an example cloud cost model for Microsoft Azure and explains how it was obtained.

In the *Determine Database Performance* task, the IOPS (Input/Output operations per second) for the source and target databases must be determined. Specifically, these values are the IOPS of the underlying storage device (i.e. SSD or HDD) the database is using. Typically cloud services will allow users to specify differing levels of storage performance in IOPS. Several methods also exist to accurately estimate an IOPS value for an existing database, such as [185].

The final task is to run the simulations using MigSim; a single simulation can only evaluate one set of migration options at a time. This will produce estimates for: migration duration, migration cost, and future cloud running costs. Detailed results, including cumulative costs and the estimated database load during the simulation, can be obtained from the MigSim log file.

5.3 Evaluation

This cloud database migration planning methodology has been evaluated in the previous thesis chapters. Section 3.3 evaluates the Planning and Modelling phases, while Section 4.4 evaluates the Planning and Simulation phases. These use a quantitative approach with a combination of case studies and experiments. In this section, an additional qualitative evaluation is performed on the methodology as a whole. This follows the well-established *Feature Analysis* method [194, 195, 196].

Feature Analysis is a comparative method for determining the most suitable methodology or tool for a given task. It identifies the requirements an organisation has, and maps these to the features a methodology/tool possesses [194]. A prioritised *feature list* is created, consisting of simple and compound features. A simple feature is one that a methodology/tool may or may not support, while a compound feature can have a variable degree of support judged on an ordinal scale. Compound features are accompanied with a 'judgement scale' to associate numerical values for some given levels of feature support. Finally, a 'assessor' determines the score for each of the candidate methodologies/tools.

5.3.1 Data

This subsection discussess the relevant properites of the methodology and its evaluation. It is structured in the same format as [196], and forms the basis for Feature Analysis.

Scope and Purpose

The cloud database migration planning methodology presented in this chapter will be compared with (1) the REMICS methodology, (2) the ARTIST methodology, and (3) the Grammar-to-Model mapping approach. These have been identified in the literature review (Chapter 2) as the leading approaches which could be applied to the challenge of planning a cloud database migration.

The literature review also identifies several tasks which organisations typically perform when planning the migration of their database to the cloud. They select a cloud provider based on cost, functionality, familiarity, or other factors. Next, they choose between a database-as-a-service or VM-based deployment. Finally, they choose between Internet-based data migration or device shipping (e.g., HDD or Amazon Snowball). This can depend on cost, time constraints, and the volume of data to be migrated.

Basis of Evaluation

The four candidate methodologies (Chapter 5, REMICS, ARTIST, and Grammar-to-Model mapping) have been evaluated using real-world databases and migrations. The planning and modelling phases of the Chapter 5 methodology included: a case study on database schemas to determine system support, a case study to measure tool extensibility, a performance experiment, and a case study to assert model completeness. Furthermore, the simulation phase used a case study to evaluate the migration duration and cost estimates. An experiment was performed to measure the accuracy of automatically scaling the database hardware in the simulation, which determines future cloud running cost estimates.

The candidate methodologies which are external to this thesis have either been included in the evaluations discussed above, or evaluated by their authors in similar case studies. This provides numerous potential metrics for judging the extent a methodology supports a feature.

Roles and Responsibilities

Multiple roles and responsibilities exist in this evaluation. The *sponsor* is Science Warehouse Ltd and other organisations who are planning the migration of their database to a cloud platform. This role is defined in [196] as the individual(s) who would invest in the chosen methodology (i.e., by applying it). Their needs will be presented as the list of features a methodology must have.

The *methodology user* role is the software or database engineers who will apply the candidate methodologies. In this case, the *evaluator* role will be performed by the author of this thesis. This involves analysing the feature list (produced from the literature review and discussions with Science Warehouse), and scoring each methodology.

Evaluation Procedure

The databases and cloud providers used for the case studies in Sections 3.3 and 4.4, were selected using several criteria. The real cloud database migrations in Section 4.4 (which evaluate the simulation phase) use the Amazon Web Services and Microsoft Azure platforms due to their market share [197, 198]. In all cases, the databases were selected so that the size of their schema and the volume of data they contain is representative. This was based on other published case studies identified in the literature review chapter.

Assumption and Constraints

The methodology is designed to support relational databases, although there is no dependency on a single database type. The application of the methodology to NoSQL database migrations is discussed in Section 6.2. Currently the DBLModeller tool, which supports the modelling phase of the methodology, requires Oracle and MySQL databases. However, this can be extend with minimal effort as shown in Section 3.3.2. Furthermore, the methodology assumes the source and target databases have the same SQL dialect (e.g., Oracle), database version (e.g., 11g), and schema. The literature review and discussions with Science Warehouse revealed that organisations will often split database migration and modernisation into separate projects.

The methodology has been evaluated quantitatively in Sections 3.3 and 4.4 primarily through case studies. Respectively, six databases were modelled and four databases were migrated to public clouds. The use of case studies means the results must be interpreted in the context of the databases they were obtained from. The database characteristics have been explained in these sections.

5.3.2 Analysis and Conclusions

A feature list for cloud database migration planning methodologies is presented below, it represents the requirements an organisation may have. The priority of the feature is indicated in brackets (either mandatory, Highly Desirable, Desirable, or Nice to have [195]), along with its type (Simple or Compound).

1. Estimation of key database migration properties.
    1.1. Data transfer duration. [Mandatory, Compound]
    1.2. Data transfer cost. [Mandatory, Compound]
    1.3. Future database running costs on a target cloud platform. [Mandatory, Compound]
2. High level of automation.
    2.1. A tool should be publicly available to automatically calculate the results when given a set of input data in the correct format. [Highly Desirable, Simple]
    2.2. A tool should be publicly available to acquire the input data from the target database and system. [Desirable, Simple]
3. High level of efficiency.
    3.1. Low estimation error. [Mandatory, Compound]
    3.2. Estimates can be obtained using a standard desktop PC. [Nice to have, Compound]
4. Compatibility with broader cloud migration methodologies. [Nice to have, Compound]

Feature 1 specifies the parameters a methodology should support when comparing different clouds. These are the minimum necessary when migrating and hosting a database with a new cloud provider. The sub-features are compound to reflect how methodologies may not support these explicitly, but may be used or adapted to include them. Features 2 to 4 are non-functional; they are included because they will typically influence an organisations decision to use a methodology.

A 'judgement scale' [195] is presented in Table 5.3 to score the compound features. For the simple features, one point is awarded if it is present in the candidate methodology. As mentioned previously, the methodology from this chapter will be compared

with REMICS, ARTIST, and the Grammar-to-Model approach. REMICS and ARTIST support the migration of software systems to the cloud. Grammar-to-Model supports the modelling of a system, although it could be used for cloud migration planning.

Table 5.3: Points accrued for different levels of feature support (compound only)

Features 1.2, 1.2, & 1.3 (Estimations)	
Supported	2
Minor Extension	1
Unsupported	0
Feature 3.1 (Relative Error)	
Very Low ($< 5\%$)	2
Acceptable ($< 6 - 25\%$)	1
Unreliable as a single source ($> 25\%$)	0
Feature 3.2 (Performance)	
Desktop	2
Single Server	1
Cluster	0
Feature 4 (Compatibility)	
Open-source and Standardised metamodels	2
Open-source or Standardised metamodels	1
Neither	0

The Cloud Database Planning Methodology fully supports the estimation of the values in Feature 1, as shown previously in Section 4.4 of this thesis. However, the other candidate methodologies have a broader scope and only briefly cover planning/feasibility. The business feasibility study in ARTIST [26] includes data transfer cost and future cloud running costs, but relies on the user to obtain values for these by other methods. REMICS [83] and the Grammar-to-Model approach [66] provide no support for cost comparison or economic feasibility. This rules-out their use here, as sub-features 1.1, 1.2, and 1.3 are mandatory. Without knowing the data transfer duration, an organisation cannot choose between HDD shipping and Internet-based data migration. Furthermore, a cost comparison between cloud providers cannot be performed without knowing data transfer and running costs.

For Feature 2, only the Cloud Database Migration Planning Methodology has complete tool support. ARTIST provides the *Migration Feasibility Assessment* and *Business Feasibility* tools. However, these rely on the user inputting correct cost values. They do not support the automatic estimation of costs from database structure and workload models.

The evaluations for ARTIST, REMICS, and Grammar-to-Model do not present accuracy results. Instead, case studies were performed to show the methodologies could be used in large-scale systems. For the Cloud Database Migration Planning Methodology, Section 4.4 shows a relative error of between 4% and 22% for estimated migration duration and cost. This was determined by comparing data from four real cloud database migrations against simulation results. The estimated future running costs were shown to have an relative error of up to 12%. This was established by deploying two systems onto multiple clouds, then measuring the database throughput with real database queries from the software. This was compared with the throughput of the model cloud database in the simulation. The maximum database throughput dictates the number of cloud database instances required, hence this is the key factor for accurate running costs.

Less variation between methodologies is seen for performance (Feature 3.2) and compatibility (Feature 4). All can be performed using a regular laptop/desktop PC, and do not require significant computing resources. Furthermore, they all used standardised metamodels (either KDM or SMM). The majority of tools were open-source, with the exception of REMICS which partially relied on commercial tools.

Table 5.4 has been produced from the feature list and the available data for each methodology. This indicates that the Cloud Database Migration Planning methodology is the most suitable solution, scoring eleven points. It also highlights that research problem RP1 (defined in Chapter 1) is not sufficiently addressed by the existing approaches.

5.4 Chapter Summary

This chapter presented a methodology for the evaluation of cloud database migration options. It brought together the DBLModeller (Chapter 3) and MigSim (Chapter 4) approaches, and explained their integration. It was developed in collaboration with Science Warehouse, and is a definition of the process followed when migrating their

Table 5.4: Feature Analysis results for each methodology

Methodology	**Score**
Cloud Database Migration Planning Methodology	11
ARTIST	8
REMICS	3
Grammar-to-Model	3

database. The methodology estimates migration cost, migration duration, and cloud running costs using models of database structure, workload, and the target cloud platform. These are crucial pieces of information when migrating a database. A detailed understanding of each allows an organisation to choose between cloud providers and different migration methods (e.g., Internet versus HDD shipping).

The methodology has three phases: planning, modelling, and simulation. Within the planning phase there are three tasks: select simulation goal, determine security requirements, and search for workload information sources. The modelling phase contains four tasks: schema extraction, obtain workload data, query logging, and model transformation. The final phase is simulation, this has the following tasks: create cloud cost model, determine database performance, and simulate. The accuracy of the results produced by the methodology has been evaluated through the DBLModeller and MigSim evaluations (Sections 3.3 and 4.4). The methodology has been published online at [11].

Chapter 6

Conclusions

This thesis presented a tool-supported methodology for the evaluation of cloud database migration options. A large number of options must be decided upon, such as: the provider, cloud instance / virtual machine specification, data migration method (Internet or device shipping), data migration security (e.g. VPN), and downtime tolerance. This is a complex process as each decision can impact the others. An incorrect decision can lead to increased costs or Service Level Agreement violations with the system users. Furthermore, choosing the best option for the system being migrated requires a fully accurate and detailed knowledge of the database and its workload. This is often not present for large, complex, or old systems.

The methodology is underpinned by a modelling (DBLModeller) and a simulation approach (MigSim). These are stand-alone solutions which may also be applied to problems other than the evaluation of cloud database migration options. For example, the models from DBLModeller can identify "hot spots" of activity with in the database which could be used to influence system design (e.g., by adding caching). The approach also measures database size and query growth, which allows organisations to determine when scaling or hardware upgrades might be required (even if they are not considering migration). The MigSim tool associates cost with resource consumption, functionality which other simulation tools do not provide. This could be applied equally when simulating software layer deployments in the cloud.

DBLModeller produces database workload and structure models from: query logs, monitoring data, and SQL schema dumps. The platform-independent models provide a detailed understanding of the historic growth trends, load fluctuations, and schema. MigSim is a discrete-event simulator which estimates the data migration cost and duration, as well as the future cloud running costs. It uses a set of standardised models

of the database structure, database workload, and target cloud pricing; the first two models are produced by DBLModeller and the cloud cost model is created manually by the user (conforming to the metamodel developed in this research).

The methodology and its components have been evaluated using several real-world systems. The results produced by MigSim were compared against the migration of two real-world systems: Apache OFBiz and Science Warehouse. These were migrated twice between Amazon Web Service and the Microsoft Azure public cloud, once in each direction. Science Warehouse were an industrial partner in this research, and 38GB of real data from their system was migrated. The Apache OFBiz database contained 200GB of synthetically generated real data.

The completeness and correctness of the models produced by DBLModeller were evaluated using four real systems: Science Warehouse, Apache OFBiz, MediaWiki, and a student record system. The time taken to extract these models was also measured over independent runs. Additionally the system support of the DBLModeller tool was evaluated using 15 real systems. Its extensibility was determined through a case study on Microsoft Sharepoint.

All the research data, source code [9] [10], and methodology [11] have been made publicly available online to enable reproducibility.

6.1 Thesis Contributions

This thesis addressed four main research problems:

RP1 No systematic method of estimating the duration of a cloud database migration exists, which is essential when selecting a migration approach and in planning its execution.

RP2 The available approaches for estimating cloud database running costs for a system rely on the user knowing future resource requirements, leading to inaccuracies.

RP3 Database modelling approaches, suitable for use in model-based migration or modernisation, are vendor-specific and target a narrow range of databases.

RP4 These approaches do not consider the cloud-relevant properties of the database structure and workload.

MigSim (Chapter 4) contributes towards solving *RP1* and *RP2*. Obtaining the methodology results (migration cost, data transfer time, and future running costs)

requires the use of simulation. The database workload and cloud performance are stochastic variables, so exact calculations are not possible. However, the existing cloud simulation tools (presented in Section 2.5) do not provide the required functionality. MigSim extends the leading cloud discrete-event simulation framework — CloudSim — to associate costs with resource consumption, auto-scaling based on load, and tracking data storage changes during runtime. This enables simulation of 'on-premise to cloud' and 'cloud to cloud' data migration using KDM structure models, SMM workload models, and cloud cost models. A cloud cost metamodel was also developed to specifically capture cloud charges incurred during database migration. As discussed in Section 4.3 no existing metamodel intuitively supported this task without extension.

The end-to-end methodology for analysing cloud database migration options (Chapter 5) also addresses *RP1* and *RP2*. Using a common notation (the Eclipse Process Framework) it specifies how DLBModeller and MigSim can be used together to compare the impact of different options on migration cost and duration. It also defines methods for capturing existing database information which must be supplied to DBLModeller. The migration options include: database instance(s) size, instance(s) type (i.e., DBaaS or a VM), cloud platform, and use of migration middleware [180]. For a particular set of options the migration cost, data transfer time, and future running costs will be estimated. This enables an organisations to systematically compare different option sets.

The contributions made by the DBLModeller approach address *RP3* and *RP4*. An inherent part of the DBLModeller design — presented in Chapter 3 — is a multi-dialect SQL grammar. This enabled support for SQL constructs used in the Oracle and MySQL schemas for 15 real-world systems. This is a significant improvement in SQL support compared to the state-of-the art approach [66], and was possible due to the reduced implementation effort a multi-dialect SQL grammar provides. However, the heterogeneity of relational databases also made it necessary to address extensibility of [66]. Achieving 100% SQL support with any approach or tool would be unfeasible. DBLModeller introduced annotated text-to-model transformations which significantly reduced the quantity of code needed to extract the models. Specifically, it allows a model-to-model transformation to the removed from the process. Section 3.3.2 showed how reducing the lines of code can improve extensibility.

DBLModeller captures the key properties required to model a database prior to cloud migration (either cloud-to-cloud or in-house to cloud). These are: required storage space, number of rows, number of columns, number of tables, short-term load patterns, and long-term growth patterns. As discussed in Chapter 2, no existing work

captured these properties together [127, 39, 4]. DBLModeller also utilises two common software re-engineering standards: the Knowledge Discovery Metamodel and the Structured Metrics Metamodel. This is essential for industry adoption and increases tool interoperability. Modelling the short-term load patterns (e.g., system load dropping overnight) enables the approach users to understand the required quantity of cloud instances and when they will be running. Modelling long-term growth trends provides a source of information when estimating future resource requirements.

The extent to which the four research problems have been addressed, has been measured in their corresponding chapters. For *RP1*, an analysis of the existing cloud migration literature was performed (Section 2.3.2). Several properties relevant to the evaluating of cloud database migration options were identified. These include short-term workload patterns, long term growth trends, required storage capacity, query types, and database structure. Together they can be used to determine the suitability of a cloud database service, but also for database re-engineering. DBLModeller captures all of these properties in standardised models. As discussed in Chapter 3, the existing work focused on partially capturing some of these properties. The lack of standardisation meant results from different approaches could not be combined. Furthermore, the successful use of the DBLModeller models by MigSim illustrates the fact that the correct properties were captured.

For *RP3* the platform-independence of DBLModeller is evaluated in terms of SQL support and extensibility. Section 3.3.1 showed that for a set of eighteen SQL schemas, DBLModeller could extract a model for each and supported 99% of the SQL keywords they contained. The previous state-of-the-art appraoch [39] could only obtain a model from a single schema. Section 3.3.2 showed that DBLModeller could be extended to support a new SQL dialect with 63% less code than [39].

6.2 Future Work

Potential areas of future work have been identified for each key output of this thesis, i.e., DBLModeller, MigSim, and the planning methodology.

1) DBLModeller

The structure and workload models produced by DBLModeller contain data that enables a variety of different analyses. For example, organisations often want to improve the database structure of the quality of data it contains, as this can bring performance and maintainability benefits [199]. The automation within this process can be increased

through model-based detection of design anti-patterns, or by detecting the use of the current database for unsuitable tasks.

Database design anti-patterns [200] [201] — like software design anti-patterns [202] — are bad coding practices which lead to non-functional problems (e.g., poor maintainability). While these anti-patterns have been identified in the literature, no approaches exist for their automated detection and removal. This presents an interesting area of future work, where some of the novel ways DBLModeller addressed database heterogeneity could be valuable. An investigation could be performed to asssess whether common database anti-patterns [201] can be identified in the models extracted by DLBModeller. Automatically detecting these at the model level is platform-independent and would assist organisations to improve the design of their database schema.

Another potential future use of DBLModeller is related to the fact that many legacy applications were designed before NoSQL data stores [203], hence relational databases were used for virtually all persistence tasks. However, for some systems moving completely to a NoSQL store or using a combination of a relatinal database and NoSQL stores (polyglot persistance [204]) may be more suitable. NoSQL data stores are usually classified as: key-value, document-based, or column-based [205]. The models extracted by DBLModeller could be used to develop an approach which automatically recommends a suitable database type(s) for a given system.

In addition to making greater use of the models DBLModeller already produces, another research area is enhancing or increasing the data they contain. Many tools and approaches existing for modelling the software layer of a system, such as MoDisco [53]. DBLModeller has the potential to integrate with these to produce a complete model of the system. The key advantage is that the relationships between the software and database layers could then be anyalsed. An organisation could check whether their intended database access method is being fully utilised, e.g., ensuring an Object-Relational Mapping library [206] is used exclusively and no hard-coded queries are present.

2) MigSim

The MigSim approach could be further developed in a variety of ways, including: the use of additional parameters when simulating database capacity, extending the approach to support relational-to-NoSQL migrations, and automating the extraction of cloud cost models. Currently, MigSim horizontally scales database capacity based on the storage/IO performance. Storage performance is often the limiting factor of database performance in online transaction processing workloads (e.g., enterprise and

web applications where user interface actions trigger database transactions to accomplish a task) [207]. However, considering CPU and RAM utilisation would increase the accuracy of the simulations. Databases with analytical or business intelligence workloads would benefit the most from this enhancement, as would databases which use complex stored procedures that make greater use of the database server's CPU.

The migration of data from a relational database to NoSQL data store is a common activity [208]. However, the process is more complex than relational-to-relational database migrations. Additional tools are typically required to transform the data; and the migration cost and duration will increase as a result. The MigSim approach could be used as-is to simulate data migration in this scenario by inducing data transformation costs in the middleware component, although the accuracy would likely be limited. Extending MigSim to consider the specific details for relational-to-NoSQL migration would increase the number organisations that could benefit from it.

A large number of public and private cloud providers exist. Currency fluctuations, the release of new services, and new hardware upgrades affect how they charge. Automating the extraction of cloud cost models would reduce error and time, so another area of future work would be the creation of a model extraction approach and tool.

In addition to these three features, an expansion of the evaluation to include several large-scale systems with database clusters would be valuable. This would confirm the findings and provide interesting data on the impact of load on migration times. Unlike Science Warehouse and OFBiz, such systems would usually require minimal or zero downtime during migration.

3) Planning Methodology

The opportunities for future work in the cloud database migration planning methodology, include: support for additional migration decisions, support for the migration process beyond planning, and further case studies.

At present the methodology allows the comparison of cloud provider, migration methods (Internet and HDD/device shipping), and migration infrastructure (e.g., VM specification). However, planning a cloud database migration involves many more steps and decisions. A significant challenge reported in [209] and by Science Warehouse is incompatibility identification between database versions. The database version an organisation is using in their on-premise datacenter may not be available on the database-as-a-service that they wish to migrate to. This introduces risk into the migration process and requires analysis and testing. Adding support for this task (or other similar tasks) would increase the industrial relevance of the methodology.

Planning and decision-making are only the first steps in migrating a database to the cloud. The methodology could be extended to guide organisations through the migration, testing, deployment, and change-over tasks. Existing cloud migration methodologies (e.g., [25] and [26]) support these tasks for the software layer only.

Appendices

Appendix A

Amazon Web Services Cost Model

```
<costs:CloudProvider xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:costs="https://www-
users.york.ac.uk/~mhe504/CloudCostsMetamodel.ecore" xmi:version="2.0"
xsi:schemaLocation="https://www-
users.york.ac.uk/~mhe504/CloudCostsMetamodel.ecore CloudCostsMetamodel.ecore"
name="Amazon Web Services">
  <utilisedService name="Database Migration Service">
    <item xsi:type="costs:ComputeCharge" name="t2.micro" unit="hour"
    type="migration">
      <costTier name="Single-AZ" lowerBound="-1" upperBound="-1" location="EU West
      (Ireland)" costPerUnit="0.020"/>
      <costTier name="Single-AZ" lowerBound="-1" upperBound="-1" location="EU West
      (London)" costPerUnit="0.021"/>
      <costTier name="Multi-AZ" lowerBound="-1" upperBound="-1" location="EU West
      (London)" costPerUnit="0.042"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="t2.small" unit="hour"
    type="migration">
      <costTier name="Single-AZ" lowerBound="-1" upperBound="-1" location="EU West
      (Ireland)" costPerUnit="0.039"/>
      <costTier name="Single-AZ" lowerBound="-1" upperBound="-1" location="EU West
      (London)" costPerUnit="0.041"/>
      <costTier name="Multi-AZ" lowerBound="-1" upperBound="-1" location="EU West
      (London)" costPerUnit="0.082"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="t2.medium" unit="hour"
    type="migration">
      <costTier name="Single-AZ" lowerBound="-1" upperBound="-1" location="EU West
      (Ireland)" costPerUnit="0.078"/>
      <costTier name="Single-AZ" lowerBound="-1" upperBound="-1" location="EU West
      (London)" costPerUnit="0.082"/>
      <costTier name="Multi-AZ" lowerBound="-1" upperBound="-1" location="EU West
      (London)" costPerUnit="0.164"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="t2.large" unit="hour"
    type="migration">
      <costTier name="Single-AZ" lowerBound="-1" upperBound="-1" location="EU West
      (Ireland)" costPerUnit="0.157"/>
      <costTier name="Single-AZ" lowerBound="-1" upperBound="-1" location="EU West
      (London)" costPerUnit="0.165"/>
      <costTier name="Multi-AZ" lowerBound="-1" upperBound="-1" location="EU West
      (London)" costPerUnit="0.33"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="c4.large" unit="hour"
    type="migration">
      <costTier name="Single-AZ" lowerBound="-1" upperBound="-1" location="EU West
      (Ireland)" costPerUnit="0.175"/>
      <costTier name="Single-AZ" lowerBound="-1" upperBound="-1" location="EU West
      (London)" costPerUnit="0.175"/>
      <costTier name="Multi-AZ" lowerBound="-1" upperBound="-1" location="EU West
      (London)" costPerUnit="0.35"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="c4.xlarge" unit="hour"
    type="migration">
      <costTier name="Single-AZ" lowerBound="-1" upperBound="-1" location="EU West
      (Ireland)" costPerUnit="0.351"/>
      <costTier name="Single-AZ" lowerBound="-1" upperBound="-1" location="EU West
      (London)" costPerUnit="0.35"/>
      <costTier name="Multi-AZ" lowerBound="-1" upperBound="-1" location="EU West
      (London)" costPerUnit="0.70"/>
    </item>
```

```
  <item xsi:type="costs:ComputeCharge" name="c4.2xlarge" unit="hour"
  type="migration">
    <costTier name="Single-AZ" lowerBound="-1" upperBound="-1" location="EU West
    (Ireland)" costPerUnit="0.703"/>
    <costTier name="Single-AZ" lowerBound="-1" upperBound="-1" location="EU West
    (London)" costPerUnit="0.701"/>
    <costTier name="Multi-AZ" lowerBound="-1" upperBound="-1" location="EU West
    (London)" costPerUnit="1.402"/>
  </item>
  <item xsi:type="costs:ComputeCharge" name="c4.4xlarge" unit="hour"
  type="migration">
    <costTier name="Single-AZ" lowerBound="-1" upperBound="-1" location="EU West
    (Ireland)" costPerUnit="1.404"/>
    <costTier name="Single-AZ" lowerBound="-1" upperBound="-1" location="EU West
    (London)" costPerUnit="1.401"/>
    <costTier name="Multi-AZ" lowerBound="-1" upperBound="-1" location="EU West
    (London)" costPerUnit="2.802"/>
  </item>
  <item xsi:type="costs:TransferCharge" name="Internet" tierUnit="gb">
    <costTier name="Free usage" upperBound="1" location="Internet"
    inboundCost="0.0" outboundCost="0.0"/>
    <costTier name="Up to 10 TB / month" lowerBound="1" upperBound="10000"
    location="Internet" inboundCost="0.0" outboundCost="0.090"/>
    <costTier name="Up to 40 TB / month" lowerBound="1" upperBound="10000"
    location="Internet" inboundCost="0.0" outboundCost="0.085"/>
    <costTier name="Up to 100 TB / month" lowerBound="1" upperBound="10000"
    location="Internet" inboundCost="0.0" outboundCost="0.070"/>
  </item>
</utilisedService>
<utilisedService name="DBaaS">
  <item xsi:type="costs:ComputeCharge" name="db.t2.micro" unit="hour" type="db">
    <costTier name="MySQL SingleAZ" lowerBound="-1" upperBound="-1" location="EU
    West (Ireland)" costPerUnit="0.018"/>
    <costTier name="Oracle SE SingleAZ" lowerBound="-1" upperBound="-1"
    location="EU West (Ireland)" costPerUnit="0.036"/>
  </item>
  <item xsi:type="costs:ComputeCharge" name="db.t2.small" unit="hour" type="db">
    <costTier name="MySQL SingleAZ" lowerBound="-1" upperBound="-1" location="EU
    West (Ireland)" costPerUnit="0.036"/>
    <costTier name="Oracle SE SingleAZ" lowerBound="-1" upperBound="-1"
    location="EU West (Ireland)" costPerUnit="0.072"/>
  </item>
  <item xsi:type="costs:ComputeCharge" name="db.t2.medium" unit="hour"
  type="db">
    <costTier name="MySQL SingleAZ" lowerBound="-1" upperBound="-1" location="EU
    West (Ireland)" costPerUnit="0.073"/>
    <costTier name="Oracle SE SingleAZ" lowerBound="-1" upperBound="-1"
    location="EU West (Ireland)" costPerUnit="0.114"/>
  </item>
  <item xsi:type="costs:ComputeCharge" name="db.t2.large" unit="hour" type="db">
    <costTier name="MySQL SingleAZ" lowerBound="-1" upperBound="-1" location="EU
    West (Ireland)" costPerUnit="0.146"/>
    <costTier name="Oracle SE SingleAZ" lowerBound="-1" upperBound="-1"
    location="EU West (Ireland)" costPerUnit="0.296"/>
  </item>
  <item xsi:type="costs:ComputeCharge" name="db.m4.large" unit="hour" type="db">
    <costTier name="MySQL SingleAZ" lowerBound="-1" upperBound="-1" location="EU
    West (Ireland)" costPerUnit="0.193"/>
    <costTier name="Oracle SE SingleAZ" lowerBound="-1" upperBound="-1"
    location="EU West (Ireland)" costPerUnit="0.460"/>
  </item>
```

```
<item xsi:type="costs:ComputeCharge" name="db.m4.xlarge" unit="hour"
type="db">
  <costTier name="MySQL SingleAZ" lowerBound="-1" upperBound="-1" location="EU
  West (Ireland)" costPerUnit="0.386"/>
  <costTier name="Oracle SE SingleAZ" lowerBound="-1" upperBound="-1"
  location="EU West (Ireland)" costPerUnit="0.920"/>
</item>
<item xsi:type="costs:ComputeCharge" name="db.m4.2xlarge" unit="hour"
type="db">
  <costTier name="MySQL SingleAZ" lowerBound="-1" upperBound="-1" location="EU
  West (Ireland)" costPerUnit="0.772"/>
  <costTier name="Oracle SE SingleAZ" lowerBound="-1" upperBound="-1"
  location="EU West (Ireland)" costPerUnit="1.840"/>
</item>
<item xsi:type="costs:ComputeCharge" name="db.m4.4xlarge" unit="hour"
type="db">
  <costTier name="MySQL SingleAZ" lowerBound="-1" upperBound="-1" location="EU
  West (Ireland)" costPerUnit="1.545"/>
  <costTier name="Oracle SE SingleAZ" lowerBound="-1" upperBound="-1"
  location="EU West (Ireland)" costPerUnit="3.679"/>
</item>
<item xsi:type="costs:ComputeCharge" name="db.m4.10xlarge" unit="hour"
type="db">
  <costTier name="MySQL SingleAZ" lowerBound="-1" upperBound="-1" location="EU
  West (Ireland)" costPerUnit="3.864"/>
</item>
<item xsi:type="costs:ComputeCharge" name="db.m3.medium" unit="hour"
type="db">
  <costTier name="MySQL SingleAZ" lowerBound="-1" upperBound="-1" location="EU
  West (Ireland)" costPerUnit="0.095"/>
  <costTier name="Oracle SE SingleAZ" lowerBound="-1" upperBound="-1"
  location="EU West (Ireland)" costPerUnit="0.245"/>
</item>
<item xsi:type="costs:ComputeCharge" name="db.m3.large" unit="hour" type="db">
  <costTier name="MySQL SingleAZ" lowerBound="-1" upperBound="-1" location="EU
  West (Ireland)" costPerUnit="0.200"/>
  <costTier name="Oracle SE SingleAZ" lowerBound="-1" upperBound="-1"
  location="EU West (Ireland)" costPerUnit="0.490"/>
</item>
<item xsi:type="costs:ComputeCharge" name="db.m3.xlarge" unit="hour"
type="db">
  <costTier name="MySQL SingleAZ" lowerBound="-1" upperBound="-1" location="EU
  West (Ireland)" costPerUnit="0.400"/>
  <costTier name="Oracle SE SingleAZ" lowerBound="-1" upperBound="-1"
  location="EU West (Ireland)" costPerUnit="0.980"/>
</item>
<item xsi:type="costs:ComputeCharge" name="db.m3.2xlarge" unit="hour"
type="db">
  <costTier name="MySQL SingleAZ" lowerBound="-1" upperBound="-1" location="EU
  West (Ireland)" costPerUnit="0.785"/>
  <costTier name="Oracle SE SingleAZ" lowerBound="-1" upperBound="-1"
  location="EU West (Ireland)" costPerUnit="1.960"/>
</item>
<item xsi:type="costs:ComputeCharge" name="db.r3.large" unit="hour" type="db">
  <costTier name="MySQL SingleAZ" lowerBound="-1" upperBound="-1" location="EU
  West (Ireland)" costPerUnit="0.265"/>
  <costTier name="Oracle SE SingleAZ" lowerBound="-1" upperBound="-1"
  location="EU West (Ireland)" costPerUnit="0.505"/>
</item>
<item xsi:type="costs:ComputeCharge" name="db.r3.xlarge" unit="hour"
type="db">
```

```
      <costTier name="MySQL SingleAZ" lowerBound="-1" upperBound="-1" location="EU
      West (Ireland)" costPerUnit="0.530"/>
      <costTier name="Oracle SE SingleAZ" lowerBound="-1" upperBound="-1"
      location="EU West (Ireland)" costPerUnit="1.010"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="db.r3.2xlarge" unit="hour"
    type="db">
      <costTier name="MySQL SingleAZ" lowerBound="-1" upperBound="-1" location="EU
      West (Ireland)" costPerUnit="1.055"/>
      <costTier name="Oracle SE SingleAZ" lowerBound="-1" upperBound="-1"
      location="EU West (Ireland)" costPerUnit="2.020"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="db.r3.4xlarge" unit="hour"
    type="db">
      <costTier name="MySQL SingleAZ" lowerBound="-1" upperBound="-1" location="EU
      West (Ireland)" costPerUnit="2.110"/>
      <costTier name="Oracle SE SingleAZ" lowerBound="-1" upperBound="-1"
      location="EU West (Ireland)" costPerUnit="4.040"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="db.r3.8xlarge" unit="hour"
    type="db">
      <costTier name="MySQL SingleAZ" lowerBound="-1" upperBound="-1" location="EU
      West (Ireland)" costPerUnit="4.215"/>
    </item>
    <item xsi:type="costs:StorageCharge" name="Magnetic Storage" unit="gb"
    unitDuration="month">
      <costTier name="GB Month" lowerBound="-1" upperBound="-1" location="EU West
      (Ireland)" costPerUnit="0.127"/>
    </item>
    <item xsi:type="costs:TransferCharge" name="Internet" tierUnit="gb">
      <costTier name="Up to 10 TB per month" upperBound="10000"
      location="Worldwide" inboundCost="0.00" outboundCost="0.09"/>
    </item>
  </utilisedService>
  <utilisedService name="EC2 Linux">
    <item xsi:type="costs:ComputeCharge" name="t2.nano" unit="hour">
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
      (Ireland)" costPerUnit="0.0063"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="t2.micro" unit="hour">
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
      (Ireland)" costPerUnit="0.013"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="t2.small" unit="hour">
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
      (Ireland)" costPerUnit="0.025"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="t2.medium" unit="hour">
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
      (Ireland)" costPerUnit="0.05"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="t2.large" unit="hour">
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
      (Ireland)" costPerUnit="0.101"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="t2.xlarge" unit="hour">
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
      (Ireland)" costPerUnit="0.202"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="t2.2xlarge" unit="hour">
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
      (Ireland)" costPerUnit="0.404"/>
```

```
</item>
<item xsi:type="costs:ComputeCharge" name="m4.large" unit="hour">
  <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
  (Ireland)" costPerUnit="0.119"/>
</item>
<item xsi:type="costs:ComputeCharge" name="m4.xlarge" unit="hour">
  <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
  (Ireland)" costPerUnit="0.238"/>
</item>
<item xsi:type="costs:ComputeCharge" name="m4.2xlarge" unit="hour">
  <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
  (Ireland)" costPerUnit="0.475"/>
</item>
<item xsi:type="costs:ComputeCharge" name="m4.4xlarge" unit="hour">
  <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
  (Ireland)" costPerUnit="0.95"/>
</item>
<item xsi:type="costs:ComputeCharge" name="m4.10xlarge" unit="hour">
  <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
  (Ireland)" costPerUnit="2.377"/>
</item>
<item xsi:type="costs:ComputeCharge" name="m4.16xlarge" unit="hour">
  <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
  (Ireland)" costPerUnit="3.803"/>
</item>
<item xsi:type="costs:ComputeCharge" name="m3.medium" unit="hour">
  <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
  (Ireland)" costPerUnit="0.073"/>
</item>
<item xsi:type="costs:ComputeCharge" name="m3.large" unit="hour">
  <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
  (Ireland)" costPerUnit="0.146"/>
</item>
<item xsi:type="costs:ComputeCharge" name="m3.xlarge" unit="hour">
  <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
  (Ireland)" costPerUnit="0.293"/>
</item>
<item xsi:type="costs:ComputeCharge" name="m3.2xlarge" unit="hour">
  <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
  (Ireland)" costPerUnit="0.585"/>
</item>
<item xsi:type="costs:ComputeCharge" name="c4.large" unit="hour">
  <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
  (Ireland)" costPerUnit="0.113"/>
</item>
<item xsi:type="costs:ComputeCharge" name="c4.xlarge" unit="hour">
  <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
  (Ireland)" costPerUnit="0.226"/>
</item>
<item xsi:type="costs:ComputeCharge" name="r3.large" unit="hour">
  <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
  (Ireland)" costPerUnit="0.185"/>
</item>
<item xsi:type="costs:ComputeCharge" name="r3.xlarge" unit="hour">
  <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
  (Ireland)" costPerUnit="0.226"/>
</item>
<item xsi:type="costs:ComputeCharge" name="i2.xlarge" unit="hour">
  <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
  (Ireland)" costPerUnit="0.938"/>
</item>
<item xsi:type="costs:ComputeCharge" name="i2.2xlarge" unit="hour">
```

```
    <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
    (Ireland)" costPerUnit="1.876"/>
  </item>
  <item xsi:type="costs:ServiceCharge" name="Unallocated Elastic IP address"
  fee="0.005" durationType="hour" durationValue="1" location="EU West
  (Ireland)"/>
 </utilisedService>
 <utilisedService name="EC2 Windows">
  <item xsi:type="costs:ComputeCharge" name="t2.nano" unit="hour">
    <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
    (Ireland)" costPerUnit="0.0086"/>
  </item>
  <item xsi:type="costs:ComputeCharge" name="t2.micro" unit="hour">
    <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
    (Ireland)" costPerUnit="0.018"/>
  </item>
  <item xsi:type="costs:ComputeCharge" name="t2.small" unit="hour">
    <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
    (Ireland)" costPerUnit="0.034"/>
  </item>
  <item xsi:type="costs:ComputeCharge" name="t2.medium" unit="hour">
    <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
    (Ireland)" costPerUnit="0.068"/>
  </item>
  <item xsi:type="costs:ComputeCharge" name="t2.large" unit="hour">
    <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
    (Ireland)" costPerUnit="0.129"/>
  </item>
  <item xsi:type="costs:ComputeCharge" name="t2.xlarge" unit="hour">
    <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
    (Ireland)" costPerUnit="0.243"/>
  </item>
  <item xsi:type="costs:ComputeCharge" name="t2.2xlarge" unit="hour">
    <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
    (Ireland)" costPerUnit="0.466"/>
  </item>
  <item xsi:type="costs:ComputeCharge" name="m4.large" unit="hour">
    <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
    (Ireland)" costPerUnit="0.211"/>
  </item>
  <item xsi:type="costs:ComputeCharge" name="m4.xlarge" unit="hour">
    <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
    (Ireland)" costPerUnit="0.422"/>
  </item>
  <item xsi:type="costs:ComputeCharge" name="m4.2xlarge" unit="hour">
    <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
    (Ireland)" costPerUnit="0.843"/>
  </item>
  <item xsi:type="costs:ComputeCharge" name="m4.4xlarge" unit="hour">
    <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
    (Ireland)" costPerUnit="1.686"/>
  </item>
  <item xsi:type="costs:ComputeCharge" name="m4.10xlarge" unit="hour">
    <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
    (Ireland)" costPerUnit="4.217"/>
  </item>
  <item xsi:type="costs:ComputeCharge" name="m4.16xlarge" unit="hour">
    <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
    (Ireland)" costPerUnit="6.747"/>
  </item>
  <item xsi:type="costs:ComputeCharge" name="m3.medium" unit="hour">
```

```
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
      (Ireland)" costPerUnit="0.129"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="m3.large" unit="hour">
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
      (Ireland)" costPerUnit="0.258"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="m3.xlarge" unit="hour">
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
      (Ireland)" costPerUnit="0.517"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="m3.2xlarge" unit="hour">
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
      (Ireland)" costPerUnit="1.033"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="c4.large" unit="hour">
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
      (Ireland)" costPerUnit="0.205"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="c4.xlarge" unit="hour">
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
      (Ireland)" costPerUnit="0.41"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="r3.large" unit="hour">
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
      (Ireland)" costPerUnit="0.29"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="r3.xlarge" unit="hour">
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
      (Ireland)" costPerUnit="0.581"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="i2.xlarge" unit="hour">
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
      (Ireland)" costPerUnit="0.973"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="i2.2xlarge" unit="hour">
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="EU West
      (Ireland)" costPerUnit="1.946"/>
    </item>
  </utilisedService>
  <utilisedService name="VPC">
    <item xsi:type="costs:ServiceCharge" name="VPN Connection" fee="0.05"
    durationType="hour" durationValue="1"/>
    <item xsi:type="costs:ServiceCharge" name="NAT Gateway" fee="0.048"
    durationType="hour" durationValue="1" location="EU West (Ireland)"/>
    <item xsi:type="costs:TransferCharge" name="NAT Gateway">
      <costTier name="GB data processed" lowerBound="-1" upperBound="-1"
      location="EU West (Ireland)" inboundCost="0.048" outboundCost="0.048"/>
    </item>
  </utilisedService>
</costs:CloudProvider>
```

Appendix B

Microsoft Azure Cost Model

```
<costs:CloudProvider xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:costs="https://www-
users.york.ac.uk/~mhe504/CloudCostsMetamodel.ecore" xmi:version="2.0"
xsi:schemaLocation="https://www-
users.york.ac.uk/~mhe504/CloudCostsMetamodel.ecore CloudCostsMetamodel.ecore"
name="Microsoft Azure">
  <utilisedService name="Windows Virtual Machines" currency="GBP">
    <item xsi:type="costs:ComputeCharge" name="A2 v2">
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="North
      Europe (Ireland)" costPerUnit="0.0969"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="A4 v2">
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="North
      Europe (Ireland)" costPerUnit="0.2042"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="A1 v2">
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="North
      Europe (Ireland)" costPerUnit="0.0134"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="A8 v2">
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="North
      Europe (Ireland)" costPerUnit="0.4286"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="A2m v2">
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="North
      Europe (Ireland)" costPerUnit="0.1565"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="A4m v2">
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="North
      Europe (Ireland)" costPerUnit="0.3287"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="A8m v2">
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="North
      Europe (Ireland)" costPerUnit="0.6902"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="D1 v2">
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="North
      Europe (Ireland)" costPerUnit="0.0909"/>
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="West
      Europe (Netherlands)" costPerUnit="0.1006"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="D2 v2">
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="North
      Europe (Ireland)" costPerUnit="0.1819"/>
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="West
      Europe (Netherlands)" costPerUnit="0.2005"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="D3 v2">
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="North
      Europe (Ireland)" costPerUnit="0.3637"/>
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="West
      Europe (Netherlands)" costPerUnit="0.4002"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="D4 v2">
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="North
      Europe (Ireland)" costPerUnit="0.7274"/>
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="West
      Europe (Netherlands)" costPerUnit="0.8005"/>
    </item>
    <item xsi:type="costs:ComputeCharge" name="D5 v2">
```

```
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="North
      Europe (Ireland)" costPerUnit="1.3848"/>
      <costTier name="On-Demand" lowerBound="-1" upperBound="-1" location="West
      Europe (Netherlands)" costPerUnit="1.5883"/>
    </item>
    <item xsi:type="costs:ServiceCharge" name="Additional Static IP" fee="0.003"
    durationType="hour" durationValue="1" location="Worldwide"/>
  </utilisedService>
  <utilisedService name="Networking">
    <item xsi:type="costs:ServiceCharge" name="Standard VPN Gateway" fee="0.1416"
    durationType="hour" durationValue="1" location="North Europe (Ireland)"/>
    <item xsi:type="costs:TransferCharge" name="Azure to Private Network"
    tierUnit="gb">
      <costTier name="First 5 GB per month " location="Worldwide" inboundCost="0"
      outboundCost="4.9999"/>
      <costTier name="5 GB to 10TB per month" lowerBound="5" upperBound="10000"
      location="Zone 1 (Europe and Americas)" inboundCost="0"
      outboundCost="0.0648"/>
    </item>
  </utilisedService>
</costs:CloudProvider>
```

Bibliography

[1] Object Management Group, "Architecture-Driven Modernization: Knowledge Discovery Meta-Model (KDM)," standard, Object Management Group, Sep 2016.

[2] Architecture-Driven Modernization Task Force, "Structured Metrics Metamodel (SMM)," 2016.

[3] "CloudML@ARTIST cost.profile." `https://github.com/artist-project/ARTIST/blob/master/source/Modeling/Cloud%20Provider%20Metamodel/CloudMl%40Artist/eu.artist.migration.tes.cloudml%40artist/supporting_profiles/cost.profile.png`, May 2015.

[4] O. Díaz, G. Puente, J. L. C. Izquierdo, and J. G. Molina, "Harvesting models from web 2.0 databases," *Software & Systems Modeling*, vol. 12, pp. 15–34, Feb 2013.

[5] M. Tighe, G. Keller, M. Bauer, and H. Lutfiyya, "DCSim: A data centre simulation tool for evaluating dynamic virtualized resource management," in *Network and service management (cnsm), 2012 8th international conference and 2012 workshop on systems virtualiztion management (svm)*, pp. 385–392, IEEE, 2012.

[6] M. Ellison, R. Calinescu, and R. F. Paige, "Towards Platform Independent Database Modelling in Enterprise Systems," in *5th International Symposium on Data to Models and Back (DataMod)*, Federation of International Conferences on Software Technologies: Applications and Foundations, pp. 42–50, Springer, 2016.

[7] M. Ellison, R. Calinescu, and R. F. Paige, "Re-engineering the Database Layer of Legacy Applications for Scalable Cloud Deployment," in *7th International Conference on Utility and Cloud Computing (Doctoral Symposium)*, pp. 976–979, IEEE, 2014.

[8] M. Ellison, R. Calinescu, and R. F. Paige, "Evaluating Cloud Database Migration Options Using Workload Models," *Journal of Cloud Computing Advances, Systems and Applications*, vol. 7, Mar 2018.

[9] M. Ellison, "DBLModeller Tool." `http://mhe504.github.io/DBLModeller`, Jul 2017.

[10] M. Ellison, "MigSim Tool." `http://mhe504.github.io/MigSim`, Jul 2017.

[11] M. Ellison, "Option Evaluation Methodology (Eclipse Process Framework)." `https://mhe504.github.io/Migration-Option-Evaluation`, Aug 2017.

[12] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[13] C. Curino, E. P. Jones, R. A. Popa, N. Malviya, E. Wu, S. Madden, H. Balakrishnan, and N. Zeldovich, "Relational Cloud: A Database-as-a-Service for the Cloud," 2011.

[14] Y. Jadeja and K. Modi, "Cloud computing-concepts, architecture and challenges," in *International Conference on Computing, Electronics and Electrical Technologies (ICCEET)*, pp. 877–880, IEEE, 2012.

[15] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, *et al.*, "Above the Clouds: A Berkeley View of Cloud Computing," Tech. Rep. UCB/EECS-2009-28, EECS Department, University of California, Berkeley, 2009.

[16] H. Sequeira, P. Carreira, T. Goldschmidt, and P. Vorst, "Energy cloud: real-time cloud-native energy management system to monitor and analyze energy consumption in multiple industrial sites," in *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, pp. 529–534, IEEE Computer Society, 2014.

[17] V. Andrikopoulos, C. Fehling, and F. Leymann, "Designing for CAP The Effect of Design Decisions on the CAP Properties of Cloudnative Applications," in *Proceedings of the 2nd International Conference on Cloud Computing and Service Science, CLOSER 2012, 1821 April 2012, Porto, Portugal*, p. 365374, SciTePress, 2012.

[18] P. Jamshidi, A. Ahmad, and C. Pahl, "Cloud Migration Research: A Systematic Review," *IEEE Transactions on Cloud Computing*, vol. 1, pp. 142–157, Oct 2013.

[19] A. Khajeh-Hosseini, D. Greenwood, and I. Sommerville, "Cloud migration: A case study of migrating an enterprise it system to IaaS," in *3rd International Conference on Cloud Computing (CLOUD)*, pp. 450–457, IEEE, 2010.

[20] S. Jones, Z. Irani, U. Sivarajah, and P. E. Love, "Risks and rewards of cloud computing in the uk public sector: A reflection on three organisational case studies," *Information Systems Frontiers*, pp. 1–24, 2017.

[21] S. Frey and W. Hasselbring, "The CloudMIG approach: Model-based migration of software systems to cloud-optimized applications," *International Journal on Advances in Software*, vol. 4, no. 3 and 4, pp. 342–353, 2011.

[22] "AWS Snowball." `https://aws.amazon.com/snowball`, 2017. Accessed: 2017-09-13.

[23] "AWS Server Migration Service." `https://aws.amazon.com/server-migration-service`, 2017. Accessed: 2017-09-19.

[24] "Azure Websites Migration Assistant." `https://azure.microsoft.com/en-gb/downloads/migration-assistant`, 2017. Accessed: 2017-09-19.

[25] P. Mohagheghi, A. J. Berre, A. Henry, F. Barbier, and A. Sadovykh, "REMICS - reuse and migration of legacy applications to interoperable cloud services," in *Towards a Service-Based Internet*, pp. 195–196, Springer, 2010.

[26] A. Menychtas, C. Santzaridou, G. Kousiouris, T. Varvarigou, L. Orue-Echevarria, J. Alonso, J. Gorronogoitia, H. Bruneliere, O. Strauss, T. Senkova, *et al.*, "ARTIST Methodology and Framework: A novel approach for the migration of legacy software on the Cloud," in *15th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pp. 424–431, IEEE, 2013.

[27] M. Stonebraker, A. Pavlo, R. Taft, and M. L. Brodie, "Enterprise database applications and the cloud: A difficult road ahead," in *Cloud Engineering (IC2E), 2014 IEEE International Conference on*, pp. 1–6, IEEE, 2014.

[28] C. Curino, E. Jones, Y. Zhang, E. Wu, and S. Madden, "Relational cloud: The case for a database service," *New England Database Summit*, pp. 1–6, 2010.

[29] S. Barker, Y. Chi, H. J. Moon, H. Hacigümüş, and P. Shenoy, "Cut me some slack: Latency-aware live migration for databases," in *Proceedings of the 15th international conference on extending database technology*, pp. 432–443, ACM, 2012.

[30] S. Date, "Should You Upload or Ship Big Data to the Cloud?," *Communications of the ACM*, vol. 59, pp. 44–51, Jun 2016.

[31] "AWS Simple Monthly Calculator." `https://calculator.s3.amazonaws.com`, 2017. Accessed: 2017-09-23.

[32] "Microsoft Azure Pricing calculator." `https://azure.microsoft.com/en-gb/pricing/calculator`, 2017. Accessed: 2017-09-23.

[33] A. Khajeh-Hosseini, D. Greenwood, J. W. Smith, and I. Sommerville, "The Cloud Adoption Toolkit: Supporting Cloud Adoption Decisions in the Enterprise," *Software: Practice and Experience*, vol. 42, no. 4, pp. 447–465, 2012.

[34] F. Leymann, C. Fehling, R. Mietzner, A. Nowak, and S. Dustdar, "Moving applications to the cloud: an approach based on application model enrichment," *International Journal of Cooperative Information Systems*, vol. 20, no. 03, pp. 307–356, 2011.

[35] R. Pérez-Castillo, I. G.-R. De Guzman, and M. Piattini, "Knowledge Discovery Metamodel-ISO/IEC 19506: A standard to modernize legacy systems," *Computer Standards & Interfaces*, vol. 33, no. 6, pp. 519–532, 2011.

[36] B. Selic, "The pragmatics of model-driven development," *IEEE software*, vol. 20, no. 5, pp. 19–25, 2003.

[37] P. Baker, S. Loh, and F. Weil, "Model-Driven Engineering in a Large Industrial Context — Motorola Case Study," *Model Driven Engineering Languages and Systems*, pp. 476–491, 2005.

[38] J. L. Cánovas Izquierdo and J. García Molina, "Extracting models from source code in software modernization," *Software and Systems Modeling*, pp. 1–22, 2012.

[39] J. L. C. Izquierdo and J. G. Molina, "An architecture-driven modernization tool for calculating metrics," *IEEE Software*, vol. 27, no. 4, pp. 37–43, 2010.

[40] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.

[41] "Architecture-Driven Modernization Task Force." `http://adm.omg.org`, 2003. Accessed: 2017-09-13.

[42] T. Mens and P. Van Gorp, "A taxonomy of model transformation," *Electronic Notes in Theoretical Computer Science*, vol. 152, pp. 125–142, 2006.

[43] M. Brambilla, J. Cabot, M. Wimmer, and L. Baresi, *Model-Driven Software Engineering in Practice: Second Edition*. Synthesis Lectures on Software Engineering, Morgan & Claypool Publishers, 2017.

[44] D. S. Kolovos, R. F. Paige, and F. Polack, "The Epsilon Object Language (EOL)," *ECMDA-FA*, vol. 6, pp. 128–142, 2006.

[45] J. Bézivin, "In search of a basic principle for model driven engineering," *Novatica Journal, Special Issue*, vol. 5, no. 2, pp. 21–24, 2004.

[46] U. Aßmann, S. Zschaler, and G. Wagner, "Ontologies, meta-models, and the model-driven paradigm," in *Ontologies for software engineering and software technology*, pp. 249–273, Springer, 2006.

[47] J. Rumbaugh, I. Jacobson, and G. Booch, *The Unified Modeling Language reference manual*. Pearson Higher Education, 2004.

[48] A. Van Deursen and P. Klint, "Domain-specific language design requires feature descriptions," *CIT. Journal of computing and information technology*, vol. 10, no. 1, pp. 1–17, 2002.

[49] W. M. Ulrich and P. Newcomb, *Information Systems Transformation: Architecture Driven Modernization Case Studies*. Burlington, MA: Morgan Kaufmann Publishers, 2010.

[50] Architecture-Driven Modernization Task Force, "Structured Patterns Metamodel Standard (SPMS)," 2015.

[51] "Information technology – Object Management Group Architecture-Driven Modernization (ADM) – Knowledge Discovery Meta-Model (KDM)," standard, International Organization for Standardization, Apr 2012.

[52] Object Management Group, "KDM Ecore implemtation." `http://omg.org/spec/KDM/1.3`, Aug 2011. Accessed: 2017-09-13.

[53] H. Bruneliere, J. Cabot, F. Jouault, and F. Madiot, "MoDisco: a generic and extensible framework for model driven reverse engineering," in *Proceedings of the IEEE/ACM international conference on Automated software engineering*, pp. 173–174, ACM, 2010.

[54] F. Fittkau, S. Frey, and W. Hasselbring, "CDOSim: Simulating cloud deployment options for software migration support," in *Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA), 2012 IEEE 6th International Workshop on the*, pp. 37–46, IEEE, 2012.

[55] A. Bergmayr, H. Bruneliere, J. L. C. Izquierdo, J. Gorronogoitia, G. Kousiouris, D. Kyriazis, P. Langer, A. Menychtas, L. Orue-Echevarria, C. Pezuela, *et al.*, "Migrating legacy software to the cloud with ARTIST," in *17th European Conference on Software Maintenance and Reengineering*, pp. 465–468, IEEE, 2013.

[56] K. Normantas, S. Sosunovas, and O. Vasilecas, "An overview of the knowledge discovery meta-model," in *Proceedings of the 13th International Conference on Computer Systems and Technologies*, pp. 52–57, ACM, 2012.

[57] D. S. M. Santibáñez, R. S. Durelli, and V. V. de Camargo, "A combined approach for concern identification in kdm models," *Journal of the Brazilian Computer Society*, vol. 21, no. 1, p. 10, 2015.

[58] B. M. Santos, R. R. Honda, R. S. Durelli, and V. V. de Camargo, "Kdm-ao: An aspect-oriented extension of the knowledge discovery metamodel," in *Software Engineering (SBES), 2014 Brazilian Symposium on*, pp. 61–70, IEEE, 2014.

[59] R. S. Durelli, D. S. Santibáñez, B. Marinho, R. Honda, M. E. Delamaro, N. Anquetil, and V. V. de Camargo, "A mapping study on architecture-driven modernization," in *Information Reuse and Integration (IRI), 2014 IEEE 15th International Conference on*, pp. 577–584, IEEE, 2014.

[60] A. J. Berre, H. De Man, and P. Lindgren, "Business model innovation with the neffics platform and vdml," in *NGEBIS2013 workshop at CAISE*, pp. 24–30, 2013.

[61] S. A. Dahab, S. Maag, and X. Che, "A software measurement framework guided by support vector machines," in *Advanced Information Networking and Applica-*

tions Workshops (WAINA), 2017 31st International Conference on, pp. 397–402, IEEE, 2017.

[62] C. Hein, F. Barbier, T. Ritter, M.-F. Wendland, and G. Dudeck, "Verification, Testing and Models@Runtime Toolkit, Interim Release," tech. rep., REMICS Consortium, Sep 2012.

[63] REMICS Consortium, *Metrino User Guide*, Apr 2013.

[64] A. Li, X. Yang, S. Kandula, and M. Zhang, "Cloudcmp: comparing public cloud providers," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pp. 1–14, ACM, 2010.

[65] E. M. Maximilien, A. Ranabahu, R. Engehausen, and L. C. Anderson, "Toward cloud-agnostic middlewares," in *Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications*, pp. 619–626, ACM, 2009.

[66] J. L. C. Izquierdo and J. G. Molina, "A Domain Specific Language for Extracting Models in Software Modernization," in *5th European Conference on Model-driven Architecture Foundations and Applications (ECMDA-FA)*, pp. 82–97, 2009.

[67] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *Grid Computing Environments Workshop*, pp. 1–10, IEEE, 2008.

[68] *The NIST Definition of Cloud Computing.* National Institute of Standards, 2011.

[69] S. Kachele, C. Spann, F. J. Hauck, and J. Domaschka, "Beyond iaas and paas: An extended cloud taxonomy for computation, storage and networking," in *Utility and Cloud Computing (UCC), 2013 IEEE/ACM 6th International Conference on*, pp. 75–82, IEEE, 2013.

[70] L. M. Vaquero, L. Rodero-Merino, and D. Morán, "Locking the sky: a survey on iaas cloud security," *Computing*, vol. 91, no. 1, pp. 93–118, 2011.

[71] D. Villegas, N. Bobroff, I. Rodero, J. Delgado, Y. Liu, A. Devarakonda, L. Fong, S. M. Sadjadi, and M. Parashar, "Cloud federation in a layered service model," *Journal of Computer and System Sciences*, vol. 78, no. 5, pp. 1330–1344, 2012.

[72] T. Dillon, C. Wu, and E. Chang, "Cloud computing: issues and challenges," in *24th IEEE International Conference on Advanced Information Networking and Applications*, pp. 27–33, Ieee, 2010.

[73] M. Kandias, N. Virvilis, and D. Gritzalis, "The insider threat in cloud computing," in *Critical Information Infrastructure Security* (S. Bologna, B. Hämmerli, D. Gritzalis, and S. Wolthusen, eds.), vol. 6983 of *Lecture Notes in Computer Science*, pp. 93–103, Springer Berlin Heidelberg, 2013.

[74] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "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*, pp. 199–212, ACM, 2009.

[75] R. Chow, P. Golle, M. Jakobsson, E. Shi, J. Staddon, R. Masuoka, and J. Molina, "Controlling data in the cloud: outsourcing computation without outsourcing control," in *Proceedings of the 2009 ACM workshop on Cloud computing security*, pp. 85–90, ACM, 2009.

[76] J. Varia, "The Total Cost of (Non) Ownership of Web Applications in the Cloud," tech. rep., Amazon Web Services, Aug 2012.

[77] M. A. Alzain, B. Soh, and E. Pardede, "MCDB: Using multi-clouds to ensure security in cloud computing," in *Ninth International Conference on Dependable, Autonomic and Secure Computing (DASC)*, pp. 784–791, IEEE, 2011.

[78] Apache, "jclouds: The Java Multi-Cloud Toolkit." `http://jclouds.apache.org/`, 2014. Accessed: 2014-03-23.

[79] Rackspace, "Openstack." `https://www.openstack.org`, 2014. Accessed: 2014-03-25.

[80] C. Low, Y. Chen, and M. Wu, "Understanding the determinants of cloud computing adoption," *Industrial management & data systems*, vol. 111, no. 7, pp. 1006–1023, 2011.

[81] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, and A. Ghalsasi, "Cloud computing – The business perspective," *Decision Support Systems*, vol. 51, pp. 176–189, Apr 2011.

[82] P. Mohagheghi and T. Sæther, "Software engineering challenges for migration to the service cloud paradigm: Ongoing work in the REMICS project," in *IEEE World Congress on Services (SERVICES)*, pp. 507–514, IEEE, 2011.

[83] G. Benguria, B. Elvesaeter, and S. Ilieva, *REMICS Handbook, Final Release*. REMICS, Jan 2013. Accessed: 2014-06-03.

[84] E. Brandtzæg, P. Mohagheghi, and S. Mosser, "Towards a domain-specific language to deploy applications in the clouds," in *CLOUD COMPUTING 2012: 3rd International Conference on Cloud Computing, GRIDs, and Virtualization*, pp. 213–218, IARIA, 2012.

[85] A. Afshar and P. Tendai, *Application Software Reengineering*. New Delhi, India: Dorling Kindersley India, 2010.

[86] ModelBus, "Fokus!mbt." `http://www.modelbus.org/modelbus/index.php/fokusmbt`, 2014. Accessed: 2014-03-09.

[87] T. Straszak and M. Smiałek, "Acceptance test generation based on detailed use case models," tech. rep., Institute of Theory of Electrical Eng, Warsaw University of Technology, 2013.

[88] C. Pezuela, "ARTIST factsheet," tech. rep., ATOS SPAIN SA, 2012.

[89] J. Whittle, J. Hutchinson, M. Rouncefield, H. Burden, and R. Heldal, "Industrial adoption of model-driven engineering: Are the tools really the problem?," in *Model-Driven Engineering Languages and Systems*, pp. 1–17, Springer, 2013.

[90] L. Orue-Echevarria, M. Escalante, and J. Alonso, "An assessment tool to prepare the leap to the cloud," in *Cloud Computing*, pp. 273–292, Springer, 2013.

[91] S. Frey and W. Hasselbring, "An extensible architecture for detecting violations of a cloud environment's constraints during legacy software system migration," in *15th European Conference on Software Maintenance and Reengineering (CSMR)*, pp. 269–278, IEEE, 2011.

[92] S. Frey, "Cloudmig xpress beta." `https://sourceforge.net/projects/cloudmigxpress/`.

[93] D. Perez-Palacin, R. Calinescu, and J. Merseguer, "Log2cloud: Log-based prediction of cost-performance trade-offs for cloud deployments," in *Proceedings of*

the 28th Annual ACM Symposium on Applied Computing, SAC '13, pp. 397–404, ACM, 2013.

[94] E. Brandtzaeg, "CloudML : A DSL for model-based realization of applications in the cloud," Master's thesis, Department of Informatics, University of Oslo, 2012.

[95] E. Brandtzæg, S. Mosser, and P. Mohagheghi, "Towards CloudML, a model-based approach to provision resources in the clouds," in *8th European Conference on Modelling Foundations and Applications*, pp. 18–27, 2012.

[96] G. Gonçalves, P. Endo, M. Santos, D. Sadok, J. Kelner, B. Melander, and J.-E. Mangs, "CloudML: an integrated language for resource, service and request description for d-clouds," in *Third International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 399–406, IEEE, 2011.

[97] J. O'Loughlin and L. Gillam, "Towards Performance Prediction for Public Infrastructure Clouds: An EC2 case study," in *5th International Conference on Cloud Computing Technology and Science (CloudCom)*, vol. 1, pp. 475–480, Dec 2013.

[98] E. Freeman and E. Freeman, *Head first design patterns*. Sebastopol, CA: O'Reilly Media, Inc, 2004.

[99] M. Chauhan and M. Babar, "Migrating Service-Oriented System to Cloud Computing: An Experience Report," in *2011 IEEE International Conference on Cloud Computing (CLOUD)*, pp. 404–411, Jul 2011.

[100] W. Teppe, "The ARNO project: Challenges and experiences in a large-scale industrial software migration project," in *13th European Conference on Software Maintenance and Reengineering*, pp. 149–158, IEEE, 2009.

[101] A. Thakar and A. Szalay, "Migrating a (large) science database to the cloud," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, HPDC '10, (New York, NY, USA), pp. 430–434, ACM, 2010.

[102] R. S. Durelli, D. S. Santibáñez, M. E. Delamaro, and V. V. de Camargo, "Towards a refactoring catalogue for knowledge discovery metamodel," in *15th International Conference on Information Reuse and Integration (IRI)*, pp. 569–576, IEEE, 2014.

[103] B. Du Bois, S. Demeyer, and J. Verelst, "Refactoring-improving coupling and cohesion of existing code," in *Proceedings of the 11th Working Conference on Reverse Engineering*, pp. 144–151, IEEE, 2004.

[104] M. Fowler, K. Beck, J. Brant, W. Opdyke, and D. Roberts, *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Object Technology Series, Pearson Education, 2012.

[105] K. Kaur and H. Singh, "Exploring design level class cohesion metrics," *Journal of Software Engineering and Applications*, vol. 3, no. 04, p. 384, 2010.

[106] J. Al Dallal and L. C. Briand, "An object-oriented high-level design-based class cohesion metric," *Information and software technology*, vol. 52, no. 12, pp. 1346–1361, 2010.

[107] G. Lewis, J. Morris, E., B. Smith, D., and S. Simanta, "SMART: Analyzing the reuse potential of legacy components in a service-oriented architecture environment," tech. rep., Software Engineering Institute, Carnegie Mellon University, Jun 2008.

[108] H. Bruneliere, J. Cabot, F. Jouault, and F. Madiot, *MoDisco User Guide: Technologies*. Eclipse, 2010.

[109] P. Sadalage and M. Fowler, *NoSQL distilled: a brief guide to the emerging world of polyglot persistence*. Upper Saddle River, NJ: Pearson Education, 2012.

[110] J. Han, E. Haihong, G. Le, and J. Du, "Survey on NoSQL database," in *6th International Conference on Pervasive Computing and Applications (ICPCA)*, pp. 363–366, IEEE, Oct 2011.

[111] I. Robinson, J. Webber, and E. Eifrem, *Graph databases*. Sebastopol, CA: O'Reilly Media, Inc, 2013.

[112] N. Leavitt, "Will NoSQL databases live up to their promise?," *Computer*, vol. 43, pp. 12–14, Feb 2010.

[113] R. Cattell, "Scalable SQL and NoSQL data stores," *ACM SIGMOD Record*, vol. 39, pp. 12–27, Dec 2010.

[114] T. Wellhausen, "Highly scalable, ultra-fast and lots of choices: A pattern approach to NoSQL," tech. rep., Tim Wellhausen consultantsy, Nov 2012.

[115] A. Lakshman and P. Malik, "Cassandra: a decentralized structured storage system," *ACM SIGOPS Operating Systems Review*, vol. 44, pp. 35–40, Apr 2010.

[116] D. Pritchett, "BASE: An ACID alternative," *ACM Queue*, vol. 6, pp. 48–55, May 2008.

[117] R. Pérez-Castillo, I. G. R. de Guzmán, D. Caivano, and M. Piattini, "Database schema elicitation to modernize relational databases," in *14th International Conference on Enterprise Information Systems*, pp. 126–132, 2012.

[118] Data Geekery GmbH, "jooq." `http://www.jooq.org/`, 2014. Accessed: 2014-04-02.

[119] Percona LLC, "Percona XtraBackup." `http://www.percona.com/software/percona-xtrabackup`, 2014. Accessed: 2014-03-20.

[120] L. Ashdown, *Oracle Database Backup and Recovery User's Guide 11g Release 2 (11.2)*. Oracle, Jul 2013. Accessed: 2014-03-20.

[121] S. Das, S. Nishimura, D. Agrawal, and A. El Abbadi, "Albatross: lightweight elasticity in shared storage databases for the cloud using live data migration," *Proceedings of the VLDB Endowment*, vol. 4, pp. 494–505, May 2011.

[122] L. Zhang, C. Wu, Z. Li, C. Guo, M. Chen, and F. Lau, "Moving big data to the cloud: An online cost-minimizing approach," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 12, pp. 2710–2721, 2013.

[123] M. Aboulsamh and J. Davies, "A formal modeling approach to information systems evolution and data migration," in *Enterprise, Business-Process and Information Systems Modeling* (T. Halpin, S. Nurcan, J. Krogstie, P. Soffer, E. Proper, R. Schmidt, and I. Bider, eds.), vol. 81 of *Lecture Notes in Business Information Processing*, pp. 383–397, Springer Berlin Heidelberg, 2011.

[124] S. Kotecha, M. Bhise, and S. Chaudhary, "Query translation for cloud databases," in *Nirma University International Conference on Engineering*, pp. 1–4, IEEE, 2011.

[125] Google, "Google cloud datastore." `https://developers.google.com/datastore`, 2014. Accessed: 2014-03-31.

[126] A. Schram and K. M. Anderson, "MySQL to NoSQL: Data modeling challenges in supporting scalability," in *Proceedings of the 3rd Annual Conference on Systems, Programming, and Applications: Software for Humanity*, SPLASH '12, pp. 191–202, ACM, 2012.

[127] J. Castrejón, G. Vargas-Solar, C. Collet, and R. Lozano, "Exschema: Discovering and maintaining schemas from polyglot persistence applications," in *29th IEEE International Conference on Software Maintenance (ICSM)*, pp. 496–499, IEEE, 2013.

[128] jccastrejon, "Exschema installation." `https://code.google.com/p/exschema/`, 2014. Accessed: 2014-03-20.

[129] P. Atzeni, F. Bugiotti, and L. Rossi, "Uniform access to non-relational database systems: The sos platform," in *Advanced Information Systems Engineering*, pp. 160–174, Springer, 2012.

[130] I. Sriram, "Speci, a simulation tool exploring cloud-scale data centres," *Cloud Computing*, pp. 381–392, 2009.

[131] D. Kliazovich, P. Bouvry, Y. Audzevich, and S. U. Khan, "GreenCloud: a packet-level simulator of energy-aware cloud computing data centers," in *Global Telecommunications Conference (GLOBECOM)*, pp. 1–5, IEEE, 2010.

[132] G. G. Castane, A. Nunez, and J. Carretero, "iCanCloud: A brief architecture overview," in *10th International Symposium on Parallel and Distributed Processing with Applications (ISPA)*, pp. 853–854, IEEE, 2012.

[133] B. Wickremasinghe, R. N. Calheiros, and R. Buyya, "CloudAnalyst: A CloudSim-based Visual Modeller for Analysing Cloud Computing Environments and Applications," in *24th International Conference on Advanced Information Networking and Applications (AINA)*, pp. 446–452, IEEE, 2010.

[134] R. N. Calheiros, M. A. Netto, C. A. De Rose, and R. Buyya, "EMUSIM: An Integrated Emulation and Simulation Environment for Modeling, Evaluation, and Validation of Performance of Cloud Computing Applications," *Software: Practice and Experience*, vol. 43, no. 5, pp. 595–612, 2013.

[135] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future generation computer systems*, vol. 28, no. 5, pp. 755–768, 2012.

[136] B. Javadi, J. Abawajy, and R. Buyya, "Failure-aware resource provisioning for hybrid cloud infrastructure," *Journal of parallel and distributed computing*, vol. 72, no. 10, pp. 1318–1331, 2012.

[137] R. N. Calheiros, R. Ranjan, and R. Buyya, "Virtual machine provisioning based on analytical performance and qos in cloud computing environments," in *Parallel processing (ICPP), 2011 international conference on*, pp. 295–304, IEEE, 2011.

[138] S. K. Garg and R. Buyya, "NetworkCloudSim: Modelling parallel applications in cloud simulations," in *4th IEEE International Conference on Utility and Cloud Computing (UCC)*, pp. 105–113, IEEE, 2011.

[139] D. Kakadia, N. Kopri, and V. Varma, "Network-aware virtual machine consolidation for large data centers," in *Proceedings of the Third International Workshop on Network-Aware Data Management*, p. 6, ACM, 2013.

[140] M. Duggan, J. Duggan, E. Howley, and E. Barrett, "An Autonomous Network Aware VM Migration Strategy in Cloud Data Centres," in *International Conference on Cloud and Autonomic Computing (ICCAC)*, pp. 24–32, IEEE, 2016.

[141] A.-F. Antonescu and T. Braun, "Simulation of SLA-based VM-scaling algorithms for cloud-distributed applications," *Future Generation Computer Systems*, vol. 54, pp. 260–273, 2016.

[142] B. Agarwal, *Basic Statistics*. New Age International Limited, 2009.

[143] K. M. Giannoutakis, A. T. Makaratzis, D. Tzovaras, C. K. Filelis-Papadopoulos, and G. A. Gravvanis, "On the power consumption modeling for the simulation of heterogeneous hpc clouds," in *Proceedings of the 1st International Workshop on Next generation of Cloud Architectures*, p. 1, ACM, 2017.

[144] L. Grosenbaugh, "Tree form: definition, interpolation, extrapolation," *The Forestry Chronicle*, vol. 42, no. 4, pp. 444–457, 1966.

[145] D. Montgomery, E. Peck, and G. Vining, *Introduction to Linear Regression Analysis*. Wiley Series in Probability and Statistics, Wiley, 2015.

[146] L. D. Schroeder, D. L. Sjoquist, and P. E. Stephan, *Understanding Regression Analysis*. Sage Publications, 1986.

[147] S. V. de Souza and R. G. Junqueira, "A procedure to assess linearity by ordinary least squares method," *Analytica Chimica Acta*, vol. 552, no. 1, pp. 25–35, 2005.

[148] R. Rhinehart, *Nonlinear Regression Modeling for Engineering Applications: Modeling, Model Validation, and Enabling Design of Experiments*. Wiley-ASME Press Series, Wiley, 2016.

[149] M. L. Johnson and S. G. Frasier, "Nonlinear least-squares analysis," *Methods in enzymology*, vol. 117, pp. 301–342, 1985.

[150] G. Kemmer and S. Keller, "Nonlinear least-squares data fitting in excel spreadsheets," *Nature protocols*, vol. 5, no. 2, p. 267, 2010.

[151] P. Teunissen, "Nonlinear least squares," *Manuscripta Geodaetica*, vol. 15, pp. 137–150, 1990.

[152] T. Kellerer, "SQL Workbench/J." `http://www.sql-workbench.net`, Feb 2017. Accessed: 2017-09-13.

[153] T. Diakoumis, "Execute Query." `http://executequery.org`, May 2017. Accessed: 2017-09-13.

[154] P. Campaniello, "The state of the open source database market: MySQL leads the way." `www.scalebase.com/the-state-of-the-open-source-database-market-mysql-leads-the-way`, 2014. Accessed: 2014-05-08.

[155] D. Feinberg, M. Adrian, and N. Heudecker, "Magic Quadrant for Operational Database Management Systems," tech. rep., Gartner, Oct 2014.

[156] Oracle, "MySQL Workbench." `www.mysql.com/products/workbench`, 2015. Accessed: 2017-09-13.

[157] Oracle, "SQL Developer." `https://www.oracle.com/technetwork/developer-tools`, Jul 2017. Accessed: 2017-09-13.

[158] Oracle, "Oracle JDBC Logging using java.util.logging," tech. rep., Sep 2009.

[159] Oracle, "The General Query Log, MySQL 5.7 Reference Manual," tech. rep., 2015.

[160] J. Canovas, "Gra2Mol: PLSQL2ASTM example project." `https://github.com/jlcanovas/gra2mol/tree/master/examples/Grammar2Model.examples.PLSQL2ASTMModel`, Sep 2014. Accessed: 2017-09-13.

[161] Object Management Group, "SMM EMOF implemtation." http://omg.org/spec/SMM/1.0, 2012. Accessed: 2017-09-13.

[162] J. J. Levandoski, P.-Å. Larson, and R. Stoica, "Identifying hot and cold data in main-memory databases," in *29th International Conference on Data Engineering*, pp. 26–37, IEEE, 2013.

[163] J. Morris, *Practical Data Migration*. British Compute Society, 2012.

[164] "P6Spy Framework." http://p6spy.github.io/p6spy, Sep 2017. Accessed: 2017-09-13.

[165] "Apache OFBiz 16.11.01." https://ofbiz.apache.org, Nov 2016. Accessed: 2017-09-13.

[166] A. Patidar, "Apache OFBiz load tests." https://github.com/hotwaxlabs/evolvingofbiz-loadtest, 2014. Accessed: 2017-09-13.

[167] A. Blake, "log4jdbc." https://github.com/arthurblake/log4jdbc, 2015. Accessed: 2017-09-13.

[168] Oracle Corporation, "MySQL 5.0 manual, keywords and reserved words." http://dev.mysql.com/doc/refman/5.0/en/keywords.html, Nov 2008. Accessed: 2017-09-13.

[169] Oracle Corporation, "Oracle 11g Release 1: Oracle Reserved Words, Keywords, and Namespaces." http://docs.oracle.com/cd/B28359_01/appdev.111/b31231/appb.htm, Sep 2008. Accessed: 2017-09-13.

[170] D. Mituzas, "Page view statistics for Wikimedia projects." http://dumps.wikimedia.org/other/pagecounts-raw, 2015. Accessed: 2015-09-16.

[171] Wikimedia, "Wikimedia Dump Index." https://dumps.wikimedia.org/backup-index.html, 2015. Accessed: 2015-09-16.

[172] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[173] R. Pérez-Castillo, J. A. Cruz-Lemus, I. G. R. de Guzmán, and M. Piattini, "A family of case studies on business process mining using MARBLE," *Journal of Systems and Software*, vol. 85, no. 6, pp. 1370–1385, 2012.

[174] W. Nowakowski, M. Smialek, A. Ambroziewicz, and T. Straszak, "Requirements-level language and tools for capturing software system essence," *Computer Science and Information Systems*, vol. 10, no. 4, pp. 1499–1524, 2013.

[175] A. Cleve, N. Noughi, and J. Hainaut, "Dynamic program analysis for database reverse engineering," in *Generative and Transformational Techniques in Software Engineering IV*, pp. 297–321, 2011.

[176] M. H. Alalfi, J. R. Cordy, and T. R. Dean, "SQL2XMI: Reverse engineering of UML-ER diagrams from relational database schemas," in *Proceedings of the 15th Working Conference on Reverse Engineering*, pp. 187–191, 2008.

[177] N. Noughi and A. Cleve, "Conceptual interpretation of SQL execution traces for program comprehension," in *6th IEEE International Workshop on Program Comprehension through Dynamic Analysis*, pp. 19–24, 2015.

[178] K. Normantas and O. Vasilecas, "Extracting term units and fact units from existing databases using the Knowledge Discovery Metamodel," *Journal of Information Science*, vol. 40, no. 4, pp. 413–425, 2014.

[179] International Organization for Standardization, "ISO/IEC 9075-11:2011 Structured Query Language," 2011.

[180] "AWS Database Migration Service." `https://aws.amazon.com/dms`, 2017. Accessed: 2017-09-13.

[181] H. Kerzner and H. R. Kerzner, *Project management: a systems approach to planning, scheduling, and controlling*. John Wiley & Sons, 2017.

[182] S. Cicmil, T. Williams, J. Thomas, and D. Hodgson, "Rethinking project management: researching the actuality of projects," *International journal of project management*, vol. 24, no. 8, pp. 675–686, 2006.

[183] C. Palmer, "Oracle Data Pump Quick Start." `http://www.oracle.com/technetwork/issue-archive/2009/09-jul/datapump11g2009-quickstart-128718.pdf`, 2009. Accessed: 2018-03-09.

[184] C. E. Dismuke and R. Lindtooth, *Ordinary Least Squares*, p. 93104. American Society of Health-System Pharmacists, 2006.

[185] J. J. Abdul Sathar Sait, "Determining the IOPS Needs for Oracle Database on AWS," tech. rep., Amazon Web Services, Dec 2014.

[186] "Science Warehouse." http://www.sciencewarehouse.com, 2017. Accessed: 2017-09-13.

[187] M. Hajjat, X. Sun, Y.-W. E. Sung, D. Maltz, S. Rao, K. Sripanidkulchai, and M. Tawarmalani, "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.

[188] S. Fogel, *Oracle Database Administrators Guide 11g Release 2*. Oracle, 2009.

[189] Amazon Web Services, "Amazon Relational Database Service: Storage for Amazon RDS." http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Storage.html, 2016. Accessed: 2016-07-01.

[190] T. Nevil, "Standard Storage and unmanaged and managed Azure VM disks." https://github.com/MicrosoftDocs/azure-docs/blob/981dbcf8ec9ee6dcd484041652efe1f980578f7b/articles/storage/storage-standard-storage.md, Apr 2017. Accessed: 2017-05-17.

[191] P. Leitner and J. Cito, "Patterns in the chaosa study of performance variation and predictability in public iaas clouds," *ACM Transactions on Internet Technology (TOIT)*, vol. 16, no. 3, p. 15, 2016.

[192] X. Li, Y. Li, T. Liu, J. Qiu, and F. Wang, "The Method and Tool of Cost Analysis for Cloud Computing," in *IEEE International Conference on Cloud Computing*, pp. 93–100, IEEE, 2009.

[193] P. Haumer, "Eclipse Process Framework Composer," *Eclipse Foundation*, 2007.

[194] B. A. Kitchenham, "Evaluating Software Engineering Methods and Tool, Part 1: The Evaluation Context and Evaluation Methods," *ACM SIGSOFT Software Engineering Notes*, vol. 21, no. 1, pp. 11–14, 1996.

[195] B. A. Kitchenham and L. Jones, "Evaluating Software Engineering Methods and Tool, Part 6: Identifying and Scoring Features," *ACM SIGSOFT Software Engineering Notes*, vol. 22, no. 2, pp. 16–18, 1997.

[196] B. A. Kitchenham, "Evaluating Software Engineering Methods and Tool, Part 7: Planning Feature Analysis Evaluation," *ACM SIGSOFT Software Engineering Notes*, vol. 22, no. 4, pp. 21–24, 1997.

[197] Gartner Inc, "Worldwide IaaS Public Cloud Services Market." https://www.gartner.com/newsroom/id/3808563, Sep 2017. Accessed: 2018-03-05.

[198] Synergy Research Group, "Cloud Growth Rate Increases." https://www.srgresearch.com/articles/cloud-growth-rate-increases-amazon-microsoft-google-all-gain-market-share, Feb 2018. Accessed: 2018-03-05.

[199] M. Riaz, E. Tempero, M. Sulayman, and E. Mendes, "Maintainability predictors for relational database-driven software applications: Extended results from a survey," *International Journal of Software Engineering and Knowledge Engineering*, vol. 23, no. 04, pp. 507–522, 2013.

[200] T.-H. Chen, W. Shang, Z. M. Jiang, A. E. Hassan, M. Nasser, and P. Flora, "Detecting performance anti-patterns for applications developed using object-relational mapping," in *Proceedings of the 36th International Conference on Software Engineering*, pp. 1001–1012, ACM, 2014.

[201] S. Ambler and P. Sadalage, *Refactoring Databases: Evolutionary Database Design*. Pearson Education, 2006.

[202] W. H. Brown, R. C. Malveau, H. W. McCormick, and T. J. Mowbray, *AntiPatterns: refactoring software, architectures, and projects in crisis*. John Wiley & Sons, Inc., 1998.

[203] A. K. Zaki, "NoSQL databases: new millennium database for big data, big users, cloud computing and its security challenges," *International Journal of Research in Engineering and Technology (IJRET)*, vol. 3, no. 15, pp. 403–409, 2014.

[204] M. Schaarschmidt, F. Gessert, and N. Ritter, "Automated Polyglot Persistence," in *Datenbanksysteme für Business, Technologie und Web*, pp. 73–82, 2015.

[205] N. Leavitt, "Will NoSQL databases live up to their promise?," *Computer*, vol. 43, no. 2, 2010.

[206] M. Keith and M. Schnicariol, "Object-relational mapping," in *Pro JPA 2*, pp. 69–106, Springer, 2009.

[207] B. Mozafari, C. Curino, A. Jindal, and S. Madden, "Performance and resource modeling in highly-concurrent OLTP workloads," in *ACM SIGMOD International Conference on Management of Data*, pp. 301–312, ACM, 2013.

[208] A. B. M. Moniruzzaman and S. A. Hossain, "NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison," *International Journal of Database Theory and Application*, vol. 6, no. 4, 2013.

[209] S. Strauch, V. Andrikopoulos, T. Bachmann, D. Karastoyanova, S. Passow, and K. Vukojevic-Haupt, "Decision support for the migration of the application database layer to the cloud," in *5th International Conference on Cloud Computing Technology and Science (CloudCom)*, vol. 1, pp. 639–646, IEEE, 2013.