Institute of Architecture of Application Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Master's Thesis Nr. 10

Refinement and Extension of the Cloud Decision Support Framework for Application Migration to the Cloud

Balduin Metz

Course of Study:	Informatik
Examiner:	Jun.-Prof. Dr.-Ing. Dimka Karastoyanova
Supervisor:	Dr. Vasilios Andrikopoulos
Commenced:	2014-08-13
Completed:	2015-02-12
CR-Classification:	D.2.7, H.3.4, H.3.5, H.4.2

Abstract

The maturity and dissemination of Cloud Computing across diverse business domains is leading to an increasing amount of migration projects with the goal to leverage the associated benefits for important legacy applications. However, the migration of applications to the cloud is a complex problem that entails various technical and organizational aspects. The existing Cloud Decision Support Framework has been a first step to provide decision makers with the means to find a suitable migration strategy. This master's thesis has refined the framework's underlying knowledge base by reviewing its decision points, decisions and their relations as well as outcomes. Based on this refinement, the framework has been extended by elaborating the relations between outcomes resulting in greater potential for decision support. In order to allow decision makers to derive migration strategies based on the framework in an interactive manner, a web application has been implemented. In a final step, an evaluation has been carried out comprising a validation of the knowledge base and, by means of a use case, a demonstration of the efficacy of the extended decision support framework.

Contents

List of Figures

List of Tables

List of Abbreviations

AHP	analytic hierarchy process	19
ANP	analytic network process	19
API	application programming interface	18
CloudDSF	Cloud Decision Support Framework	14
CloudDSF Prototype	Cloud Decision Support Framework Prototype	14
CloudDSF+ Parser	Cloud Decision Support Framework Plus Parser	98
CloudDSF+ Prototype	Cloud Decision Support Framework Plus Prototype	95
CSS	Cascading Style Sheets	96
D3	Data-Driven Documents	97
DSS	decision support system	19
HTML	Hypertext Markup Language	96
IaaS	infrastructure as a service	16
IT	information technology	112
JS	JavaScript	96
JSON	JavaScript Object Notation	95
MCDM	multiple-criteria decision making	19
MTA	multi-tenant architecture	41
MTL	multi-tenancy level	41
OS	operating system	38
PaaS	platform as a service	16
QoS	quality of service	17
SaaS	software as a service	16
SLA	service level agreement	17
SVG	Scalable Vector Graphics	99
VM	virtual machine	33

1. Introduction

The ubiquitous presence of cloud computing in information technology is evident and its application will continue to grow significantly in the foreseeable future [1], [2], [3], [4]. Characteristics of cloud computing, e.g. on-demand provisioning of computing resources in a self-service manner according to current needs and new pricing models such as pay-per-use, are increasing company flexibility while lowering their costs at the same time [5], [6], [7]. Naturally, companies try to leverage the benefits of cloud computing, not only for their new but also for their existing legacy applications by migrating them into the cloud [8], [9].

The biggest obstacles for migrating applications to the cloud have been and still are security related issues, especially in respect to sensitive business data [5], [6], [10]. However, those concerns are decreasing as cloud computing is getting more and more mature and has proved its viability [11]. Hybrid clouds (i.e. part of the application is running on-premise and part in an off-premise cloud computing environment) and platform services (e.g. application development and middleware capabilities) are gaining more and more popularity [3]. As a consequence, it is estimated that companies will migrate not only more of their information technology infrastructure but also highly customized and important applications such as enterprise resource planning systems into the cloud [3], [12].

Moving the whole application via virtualization technology into the cloud is a method with minimal invasiveness in regard to the application and has been popular in the past [13]. To fully leverage the benefits of cloud computing, legacy applications have to be migrated in a more sophisticated granular manner in order to adapt them to their most suiting cloud computing environment [13], [14]. Toward this goal, cloud migration practices today are more mature and have significantly improved [15]. The decisions of which parts of an application and how to migrate them to the cloud or whether to migrate at all, form a multi dimensional problem that spans various technical as well as non-technical domains [15], [16]. Even though there are several different approaches supporting decision makers in moving their application into the cloud, none of those can be considered fully-fledged leading to the necessity of further research in that area [15].

In [17] a conceptual view of the decisions and tasks necessary to be considered when migrating an application into the cloud has been developed – the Cloud Decision Support Framework (CloudDSF). The goal of CloudDSF is to support decision makers "in evaluating the need for migration, and guide them along the decisions that need to be made before the actual migration process" [17, p. 1]. CloudDSF defines ten tasks and four main decision points which subsume multiple decisions which in turn subsume multiple outcomes. The decisions are strongly interconnected and form a dense network of relationships. A prototypical implementation of CloudDSF, the Cloud Decision Support Framework Prototype (CloudDSF Prototype)[1] has been developed to offer an easy comprehensible visualization of the framework and the relations between its components [17].

Currently, the relationships in the CloudDSF Prototype are only defined on the level of decisions and not on the level of concrete outcomes. Towards a more sophisticated decision support framework, the relationships have to be further refined and visualized as stated in [17]. To that end, this master's thesis builds upon the CloudDSF and covers the following research objectives:

1. Update of the state of the art in decision support systems for application migration to the cloud.
2. Refinement of the CloudDSF knowledge base and review of the relations between decisions defined in [18].
3. Extension of the decision support framework by elaborating and defining the relations between outcomes based on the previous task.
4. Enhancement of the functionality offered by the CloudDSF Prototype by reflecting, encoding and identifying an appropriate visualization mechanism for the previously defined extended decision support framework.
5. Evaluation of the extended decision support framework with respect to its knowledge base and the relations between decisions and/or outcomes.

As mentioned before, CloudDSF defines ten tasks that are necessary to support the decision making process with respect to the decision points. The refinement of those tasks and their relationships are *excluded* from the scope of this master's thesis and left for future work.

The outline of this thesis is aligned to the research objectives stated above. Chapter 2 shortly introduces the related work this thesis is based on, gives a summary of the state of the art in decision support for application migration to the cloud and

[1] The implementation can be accessed at `http://www.clouddsf.com`.

describes the current state of CloudDSF in more detail. The results of the refinement of the CloudDSF knowledge base and the relations between decisions are stated in Chapter 3. The extension of the decision support framework by defining the relationships between outcomes, is covered in Chapter 4. Chapter 5 addresses the appropriate visualization mechanism for the extended decision support framework and the enhancement of the existing CloudDSF Prototype. In Chapter 6 an evaluation of the updated CloudDSF is conducted. The thesis concludes in Chapter 7 with a summary and gives a brief overview of further needed research and future work.

2. Related Work

In the following chapter the definition of cloud computing used in this thesis is stated and the motivations as well as obstacles for application migration to the cloud are described. Subsequently, the state of the art in decision support for application migration to the cloud will be summarized. Finally, the current state of CloudDSF will be described in detail.

2.1. Cloud Computing

This thesis is a follow-up work of [17] and [18], hence the same well accepted definition of cloud computing from the National Institute of Standard and Technology will be used. For the sake of completeness the definition is stated in the following.

> "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models." [19, p. 2]

Further information regarding the five essential characteristics (i.e. on-demand self-service, broad network access, resource pooling, rapid elasticity, measured service), the three service models (i.e. infrastructure as a service (IaaS), platform as a service (PaaS), software as a service (SaaS)) and the four deployment models (i.e. public/-community/private/hybrid cloud) can be obtained, amongst others sources, from [10], [19], [20].

In the cloud computing business two essential actors can be distinguished: the *cloud provider* and the *cloud consumer* [10]. The cloud provider offers a cloud service towards a cloud consumer who logically consumes the offered service. It is critical to mention that a cloud provider can in turn consume services from other cloud

providers. Therefore, a cloud provider can be a cloud consumer at the same time and vice versa.

The utilization of cloud offerings through the cloud consumer is usually legally determined by a contract including a service level agreement (SLA). The SLA states promises and limitations of the provider including remedies in case of performance failures and obligations for the consumer [10]. Subsequently, the terms provider and consumer will be used interchangeably with cloud provider and cloud consumer respectively except when explicitly stated otherwise.

2.2. Application Migration to the Cloud

Migrating a software application can be considered as a special form of software maintenance since the goal is to maintain the functionality of the software in a different or changing operating environment [21]. *Application migration to the cloud can therefore be described as moving an application from a local data center into a cloud environment without changing its functionality or decreasing its performance* [14]. Throughout this work this understanding of application migration to the cloud will be used.

The motivation of companies to migrate a legacy application into the cloud corresponds to the associated benefits of cloud computing in general. In [3] an overview of the most often leveraged benefits, such as cost savings, greater scalability and faster access to infrastructure are stated. One of the main benefits, especially for first time cloud users, is the transformation of capital expenses to operational expenses by procuring the computing resources directly from a cloud provider instead of providing them on-premise [22], [23]. Due to the economics of scale and highly specialized knowledge, cloud providers can maintain and offer those resources at far lower prices [7], [24]. The consumers are also freed from the burden of low-level tasks such as infrastructure management which enables them to focus on more important business related activities [24].

Moreover, applications in the cloud can rapidly adapt the provisioned resources to changing workloads avoiding over- or under-provisioning, hence reducing costs and the risk of losing customers due to poor quality of service (QoS) or low availability [5], [22]. The same potential for adaptation is applicable to license management. Licenses can be payed in a pay-as-you-go manner instead of buying a fixed amount of nonrefundable licenses that in hindsight may turn out to be underutilized or not necessary at all [5], [23]. Furthermore, through the self-service manner of cloud

computing consumers are able to access infrastructure and standardized platforms for application development/testing and deployment in a fast and flexible manner [6].

On the other hand, the challenges and risks that come with cloud computing are discouraging companies to migrate their applications. Among them are security related issues which are by far the major obstacles to the adoption of cloud computing [3], [5], [6], [23]. Moving applications and data into the cloud poses new security threats due to new prospective points of attack which do not exist or cannot be mitigated as in traditional on-premise IT environments. The nine most serious threats to cloud consumers and providers alike are described by the Cloud Security Alliance and are as follows in descending order of the threat level: data breaches, data loss, account hijacking, insecure application programming interfaces (APIs), denial of service, malicious insiders, abuse of cloud services, insufficient due diligence, shared technology issues [25].

Relating to security, first and foremost companies are concerned about their sensitive business data. Those concerns are legitimate through the aforementioned threats and a lack of transparency regarding how and where providers store customer data and which security measures they implement to prevent unauthorized access [5], [10], [20], [24]. In addition, relying on a single cloud provider can lead to a vendor lock-in due to the lack of common standards between cloud providers increasing the vulnerability towards outages [5].

Further obstacles are business continuity, compliance, data transfer bottlenecks, integration to internal systems, lack of expertise and the difficulties of the management of multiple cloud services [3], [5]. However, the survey in [3] states that the benefits of cloud computing are increasing significantly according to the company's level of expertise in cloud computing. Simultaneously, the challenges are decreasing sharply, especially for security related issues [3].

This concurs with the previously mentioned higher adoption rate by companies of cloud computing in general and the migration of more important applications. As mentioned in Chapter 1, companies prefer hybrid cloud models to exploit the benefits of cloud computing while minimizing the number and severity of security threats [2], [3]. A hybrid cloud can also be used for cloud bursting, meaning if local computing resources are insufficient the application transfers workload seamlessly into the cloud [10]. Thereby performance can be ensured and/or costs can be reduced. However, hybrid cloud scenarios are often very complex and more difficult to implement in contrast to private or public clouds [10], [20].

It is obvious that legacy applications have to be adjusted in order to exploit a cloud environment and depending on the migration scenario different reengineering efforts have to be invested [13], [26]. Due to the high complexity of cloud architectures and of the existing applications, respectively, an assessment prior to a migration should be conducted to determine if a migration would be beneficial or not [15], [23]. Architectural constraints, such as special hardware or safety-criticality can be pivotal factors that inhibit a migration [10], [23]. Other factors which have to be considered before migrating applications can be non-technical e.g. organizational, legal, compliance-related, financial or technical e.g. existing infrastructure, IT skills, application architecture [15].

2.3. Application Migration and Decision Support

The complexity of the task, the high amount of implementation possibilities and decisions, and the transforming effect on the business make decision support regarding application migration to the cloud necessary. Decision support for a specific problem can be realized through a decision support system (DSS). A detailed explanation about decision support, DSSs and their architecture can be found in [27] as well as [28]. The latter also includes a distinction of DSSs into five different types. One of those types is knowledge-driven DSSs. Those knowledge-driven DSSs suggest and recommend actions to the user based on a knowledge base about the specific problem domain. This type is applicable to CloudDSF, its knowledge base and would also be appropriate to select a cloud provider for an application migration [18].

As explained above, migrating an application to the cloud spans multiple decisions which relate to, and influence each other and include qualitative and quantitative criteria. It can be therefore classified as a multiple-criteria decision making (MCDM) problem [18], [29]. To solve MCDM problems several approaches are available [29]. Some of them with respect to CloudDSF have been briefly described in [18]. It must be mentioned that the goal of CloudDSF at this stage is not to automatically determine a single optimized solution of the migration problem, but rather to support the decision making process by visualizing the necessary decisions and their relationships.

In order to solve MCDM problems, techniques like analytic hierarchy process (AHP) and analytic network process (ANP) can be applied. The AHP method is preferred in research, see Section 2.4, mainly due to lower complexity. However, AHP comes with challenges that hamper the application for decision support for application migration to the cloud. The criteria have to be brought into a hierarchical order which is often

not possible since several criteria might be related and the alternatives may affect criteria across levels. Furthermore, many different hierarchies could be constructed especially if the decisions are numerous, possibly yielding different decision results. As a consequence, various methods are applied by the recent approaches for decision support for application migration to the cloud.

2.4. State of the Art in Decision Support for Application Migration to the Cloud

The summary of the state of the art in application migration to the cloud will be partly based on previous work. More specifically, in 2013 Jamshidi, Ahmad, and Pahl conducted an extensive literature review regarding decision support for cloud migration in general [15]. Approaches specifically related to CloudDSF have been identified in [16], [17], [18]. The evaluation of CloudDSF is based on the approaches and frameworks stated in the aforementioned literature [17]. Therefore, it is assumed that relevant decision support frameworks for application migration to the cloud prior to 2013 have been covered.

In Table 2.1 the most relevant approaches to CloudDSF are identified. The first five have already been discussed in [18] including a summary of each approach pointing out its method as well as deficiencies. Further information about an approach can be obtained from its respective reference. For the purpose of this work, each one of the first five approaches have been reviewed regarding their announced plans for future work, follow-up publications from the contributing authors and relevant publications citing the particular approach.[1]

Only for one approach, namely *CloudGenius*, a directly related follow-up work has been published. In [34] the *CloudGenius* framework has been extended to support the migrations of multi-component web applications. The goal is to help engineers to select the best service mix at the IaaS layer and to enable the migration of web application clusters to cloud infrastructures. For that purpose Menzel et al. present a multi-criteria-based selection algorithm based on AHP [34]. They implemented an algorithm which is based on a genetic approach due to the rising complexity, in their *CumulusGenius* prototype.[2]

[1] Cross-citations and follow-up publications have been searched with ACM: `http://dl.acm.org`, IEEE: `http://ieeexplore.ieee.org` and Google Scholar: `http://scholar.google.de`.

[2] The prototype is available at `http://cumulusgenius.appspot.com`.

Table 2.1.: State of the art approaches in decision support for application migration to the cloud, partly based on [18] and [17].

Name	Year	Reference(s)	Method
Cloudward Bound	2010	[30]	Integer Linear Programming
The Cloud Adoption Toolkit	2012	[31]	Checklist
Cloudstep	2012	[32]	Question-Based
CloudGenius	2012	[33], [34]	AHP
Towards Process Support for Migrating Applications to Cloud Computing	2012	[35]	Step-Based
ARTIST	2013	[36], [37]	Model-Driven
CloudMIG	2013	[38]	Architecture/Model-Driven
Moving Business Intelligence to Cloud Environments (InCLOUDer)	2014	[39], [40], [41]	AHP
Legacy-to-Cloud Migration Horseshoe	2014	[42]	Architecture-Driven

For *Cloudstep* there have been no further publications and the planned prototype is not yet available. *Cloudward Bound* was only cited by a few papers dealing with very specific issues such as live migration of virtual machine images or middleware capabilities in the cloud. The cost modeling component of the *Cloud Adoption Toolkit* was acquired by *RightScale* and is now available under `www.planforcloud.com`. The other parts of the toolkit have yet to be further developed. For the approach from Chauhan and Babar no follow-up work has been published but one of the authors contributed to a new framework that will be described in more detail later on.

In order to identify new approaches relevant to CloudDSF a literature search including secondary literature [43], [44], [45] has been conducted and the outcomes of the aforementioned reviews have been used. In the following each new approach, also identified in Table 2.1, will be briefly summarized.

The *ARTIST* framework [36] suggests a software modernization approach covering all aspects and processes including the necessary tools to support the complete migration process. The added value of the framework is asserted to be as follows: a feasibility analysis prior to any investment, focus on cloud-compliant architecture issues, consideration of business aspects that are strongly linked to the technical decisions including the migration impact on the organization, fostering of reusability and the preparation for an evolution of the migrated application [36]. The tools are mainly based on the open source Eclipse Modeling Tools IDE.

Four major phases are defined which are further refined through tasks and disciplines. Firstly, in the *pre-migration* phase a gap analysis in form of a business and technical feasibility analysis is conducted to determine if a migration is beneficial and the consequences of possible migration strategies are analyzed. A more detailed view on this phase with a theoretical exercise can be found in [37]. Secondly, based on the previous results, a customized *migration* phase is following performing the actual migration. Through reverse engineering a platform-specific model is created, consolidated and then transformed into a platform-independent model to leverage the benefits of patterns across several modernization scenarios. The application elements are examined and profiled regarding performance and usage characteristics to define the necessary target environment. The platform-independent model is then transformed, with regard to the requirements of the target cloud environment, into a cloud-specific model which is than transformed via forward engineering into the executable migrated software. Thirdly, in the *post-migration* phase the application is deployed and verified in order to assure that the defined goals of the migration have been achieved. Finally, in the *migration artifact reuse and evolution phase* needed maintenance activities such as updates or cloud provider changes are performed. Furthermore, artifacts produced along the migration process that can be reused across projects are made available via a marketplace.

The *Legacy-to-Cloud Migration Horseshoe* from Ahmad and Babar extends the classical reengineering horseshoe model and the OMG's Architecture Driven Modernization framework[3] to develop a framework that provides a process-driven solution for legacy migration to the cloud [42]. The framework consists of four main processes with several fine-grained subprocesses and activities, as can be seen in Fig. 2.1, thus allowing an incremental step-by-step migration. The framework enables round-trip engineering by using the legacy source code and transforming it into cloud-enabled code. Afterwards, the migrated code can be again used as input and further refined. In comparison to *ARTIST*, the authors state that their approach is less comprehensive by excluding the deployment and certification process which is why less efforts are required to enable a migration [42].

Juan-Verdejo and Baars first proposed a migration framework in [39] with focus on partially moving applications to the cloud in the field of business intelligence. Due to the complexity of business intelligence applications, they argue that their approach can also be generalized for other applications. They take a decision support approach for the creation of architectures that, based on an iterative selection of cloud-based components, combines the local and cloud infrastructure. Compared to other approaches, the authors see the difference of their work in a holistic ap-

[3] Object Management Group: `http://www.omg.org/adm`

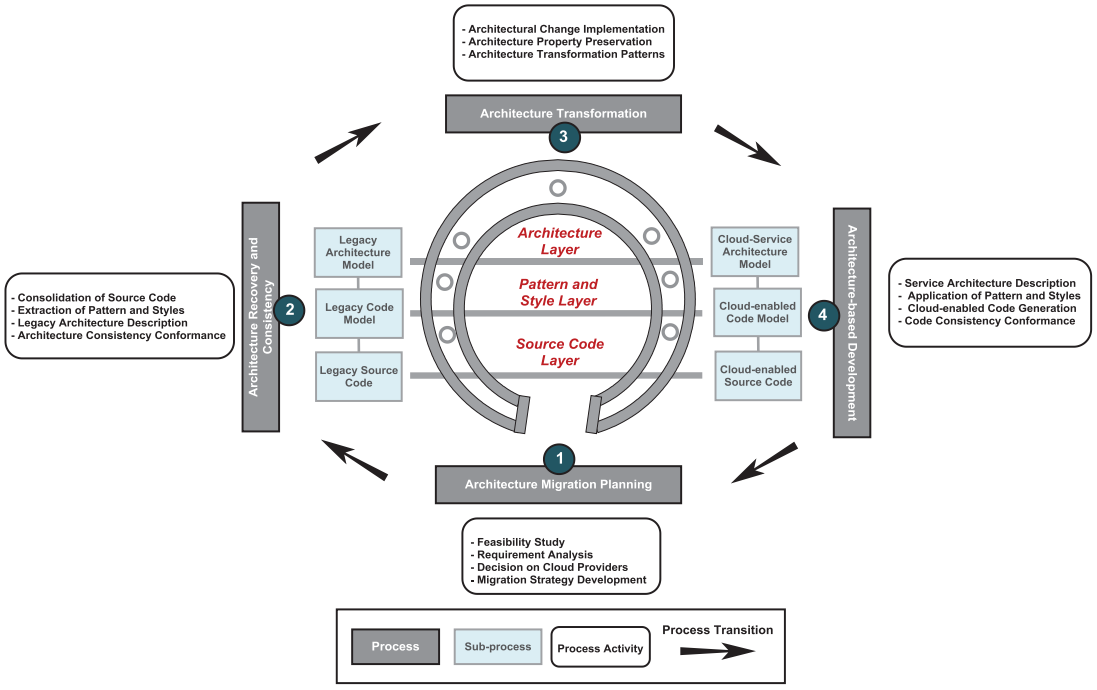

Figure 2.1.: Overview of the Legacy-to-Cloud Migration Horseshoe Framework [42].

proach by taking into account many interdependent parameters. “These parameters include business and economic considerations, technical and security-related challenges, and the organizational implications of the partial adoption of cloud computing as computation model” [39, p. 37].

In their approach three major steps can be distinguished beginning with the architectural description of the existing system. The relations between components, their properties, the users and the transactions between them are modeled to explicitly state the dependencies of the application. In the second step, migration alternatives are generated based on a model for component placement with the goal of maximizing cost benefits while respecting the requirements (maximization problem). In the last step a decision between the alternatives is made via AHP. To that end the problem is decomposed, hierarchically structured, and the relevant criteria are selected. Afterwards, those criteria are prioritized, the overall score for each alternative is calculated, and finally the best solution is chosen. A detailed explanation can be found in [41] where a spin-off of the method is presented as a stand-alone formalized decision support modeling approach to migrate application to cloud environments called *InClouder*.

More specifically in [40], the previously described framework has been extended

and a prototype based on the Eclipse Rich Client Platform and EMF-based Modeling[4] has been implemented. It consists of a system and a cloud environment modeling module which delivers the inputs for the migration strategies generator. This generator uses a database with stored migration strategies and includes a synchronization as well as a security module to incorporate consistency and data protection, e.g. encryption, policies and tokenization directly into the migrated cloud solution.

The approach *CloudMIG* aims to support the semi-automatic migration of applications to IaaS and PaaS-based environments [38]. The corresponding prototype *CloudMIG Xpress*[5] offers various features such as extracting code models from the software, estimation and comparison of costs between deployment options and detection of code violations with respect to defined cloud environment constraints. The framework has been continuously developed yielding several tools and research works[6]. The *CloudMIG* approach makes use of architecture-driven, optimization, simulation and semantic modeling methods and delivers a set of cloud deployment options from which the user can choose the most suitable alternative [38], [43].

2.5. Cloud Decision Support Framework

Andrikopoulos, Strauch, and Leymann first introduced a conceptual framework for application migration to the cloud in [16], envisioning a decision support framework that sees migration to the cloud as a multi-dimensional problem with multiple correlating decision points and related analysis tasks as can be seen in Fig. 2.2.

At that time, other work often considered only one specific kind of migration decision, e.g., selecting the right cloud provider for IaaS. They argued that through new cloud provider offerings various alternative migration scenarios are possible. Hence, decision support needs to include options for partial distribution of an application, scaling strategies and implementation of multi-tenancy.

To overcome the initial deficiency of coarse grained decision points lacking concrete decisions and outcomes to actually support decision making, an elaborated version of the framework, the aforementioned Cloud Decision Support Framework (CloudDSF), has been developed [17], [18]. CloudDSF is based on a knowledge base that contains for each decision point several decisions that in turn subsume multiple outcomes. Table 2.2 provides an overview of the knowledge base. In the following

[4] http://www.eclipse.org/modeling/emf
[5] http://sourceforge.net/projects/cloudmigxpress
[6] http://sourceforge.net/p/cloudmigxpress/wiki/Publications

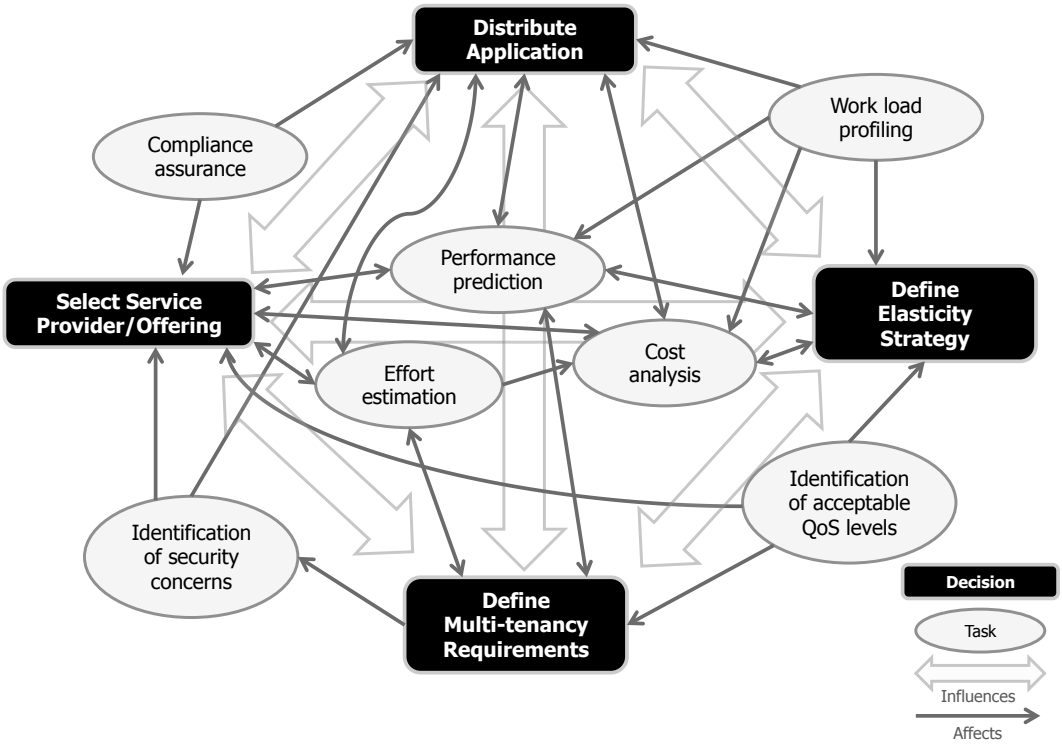

Figure 2.2.: Conceptual view of the decision support framework for cloud migration [16].

sections the decisions points, the associated tasks and the prototypical implementation of CloudDSF will be explained in more detail.

Table 2.2.: The CloudDSF knowledge base showing all decision points, decisions and outcomes [17].

Decision Point – Define Application Distribution	
Select Application Layer	Presentation/Business/Data Layer Multiple Layers
Select Application Tier	Client/Application/Data Tier Multiple Tiers
Select Application Components	Single Component Multiple Components
Select Migration Type	Type I/II/III/IV

Decision Point – Define Elasticity Strategy	
Define Scalability Level	Instance/Container/VM/Virtual Resource/Physical Hardware Level Multiple Levels
Select Scaling Type	Vertical Scaling Horizontal Scaling Hybrid Scaling
Select Elasticity Automation Degree	Manual Scaling Semi-Automatic Scaling Automatic Scaling
Select Scaling Trigger	Event-Driven/Proactive
Decision Point – Define Multi-Tenancy Requirements	
Select Kind of Multi-Tenancy	Multiple Instances Multi-Tenancy Native Multi-Tenancy
Select Multi-Tenancy Architecture	Any of the 29 Possible Combinations
Decision Point – Select Service Provider / Offering	
Select Cloud Deployment Model	Public Cloud Private Cloud Community Cloud Hybrid Cloud
Select Cloud Service Model	IaaS/PaaS/SaaS
Define Cloud Hosting	On-Premise Hosting Off-Premise Hosting Hybrid Hosting
Define Roles of Responsibility	Ownership/Operation/Management Role Any Combination of Roles
Select Cloud Vendor	Evaluated Cloud Vendor
Select Pricing Model	Free Pay-Per-Use Pay-Per-Unit Charge-Per-Use (Subscription) Combined Pricing Model
Define Resource Location	Evaluated Physical Resource Location

2.5.1. Decision Points

There are four main decision points in the CloudDSF that are briefly described in the following based on the definitions in [16], [17], [18]. In order to refine the relationships later on, a more detailed view on each decision point will be provided in Chapter 3 and Chapter 4.

- **Decision Point – Application Distribution:** The distribution of the application, i.e., which parts should be migrated to the cloud is an essential decision. Distribution can be decided based on logical layers, physical tiers or components probably spanning multiple layers and tiers. Furthermore, one of the four migration type as defined in [13] can be used.

- **Decision Point – Define Elasticity Strategy:** One of the major benefits associated with cloud computing is the wide range of scaling possibilities. Hence, this decision point is concerned with the decisions which component(s) and on what level they have to be scaled. Further decisions are what kind of scaling type as well as trigger should be used and the desired extend of automation.

- **Decision Point – Define Multi-Tenancy Requirements:** Serving multiple tenants from a common pool of resources while separating them in terms of communication (isolation of message exchanges) and administration (tenant-specific configurations) is an important aspect of cloud computing. Multi-tenancy can be realized on different system levels and can range, for example from, isolation based on an application instance and a separate database schema, to a single virtual machine per tenant on the virtualization level. Therefore, several possibilities regarding multi-tenancy must be considered.

- **Decision Point – Select Service Provider / Offering:** This decision point covers fundamental decisions regarding a migration e.g. deployment/service model and cloud hosting that have a severe impact on other decisions. Moreover, decisions with a stronger organizational view, such as the definition of roles of responsibilities, pricing models, the selection of a particular cloud vendor and where the resources have to be located geographically are also part of this decision point.

2.5.2. Analysis Tasks

Seven tasks are defined and described in [16] and three additional tasks have been added in [18] leading to a total of ten tasks in the current CloudDSF. Even though they are not in the scope of this thesis, a short explanation for each of them will be given to foster the understanding of the framework. Moreover, they might be referred to in the subsequent chapters for explanatory purposes.

- **Workload Profiling:** Determining or estimating the workload profile the application will have to cope with is an important task directly influencing the decisions how to distribute an application and which elasticity strategy to use. The results of this task are used as input for performance calculation and cost analysis.
- **Cost Analysis:** Long-time operational costs as well as one-time expenses for the migration have to be considered to decide if a migration is beneficial from a monetary point of view. Costs are influenced by the application distribution, service provider/offering selection and elasticity strategy decisions. Furthermore, workload profiling and effort estimation and thus indirectly multi-tenancy requirements, influence the cost analysis task as well. The strong interdependencies between those tasks and a possible discrepancy between estimated and actual costs lead inevitably to adjustments and feedback loops.
- **Effort Estimation:** The amount of work required to adapt the application must be estimated. Input is required from the application distribution, service provider selection and multi-tenancy requirements decisions since they all influence the amount of adaptations. As a result, those decisions could be changed to decrease the needed effort, or less likely to increase it in order to realize a more sophisticated migration.
- **Performance Prediction:** Projections of the nonfunctional behavior of the application after it is migrated to the cloud is essential to avoid a performance degrading implementation. The performance is influenced by the application distribution, selected service provider/offering and the elasticity strategy. Input can be provided by the workload profiling task to estimate performance metrics.
- **Identification of Acceptable QoS Levels:** Existing and planned SLAs can be used to infer the needed level of QoS e.g. in terms of availability. The defined QoS metrics can affect service provider/offering selection and guides the definition of a suitable elasticity strategy while constraining the options for application multi-tenancy.

- **Compliance Assurance:** A migrated application has to remain compliant to external and internal regulations. Data privacy regulations can influence or even determine the distribution of the application and therefore also the possible cloud providers and offerings.
- **Identification of Security Concerns:** In Section 2.2 several risks and threats related to cloud computing have been stated. Data and communications that are critical to protect have to be examined regarding those threats.
- **Workforce Capabilities Identification:** Using cloud services, particularly in the case of private and hybrid clouds, can impose new roles, tasks and needed skills on the cloud consumer. An identification of the workforce capabilities is therefore necessary to identify skill deficiencies. Based on the revealed deficiencies costs for training or knowledge acquisitions can be estimated more precisely and migration decisions can be adapted. The influenced decisions are mainly related to cloud hosting, roles of responsibility and the elasticity automation degree.
- **Application Analysis:** Proper decision making regarding application distribution can only be made based on a detailed view of the current state of the application. Therefore, an analysis of the existing application and its characteristics (e.g. architecture, programming language, current hardware) is necessary. Workload profiling, performance prediction and the identification of acceptable QoS levels are related to this task.
- **Vendor Benchmark:** While selecting a cloud vendor one has to consider organization-specific aspects such as reputation (e.g. reference projects, benchmarks) and capabilities (e.g. knowledge, technical skills). The task's emphasis on nontechnical aspects is intentional in order to complement the mostly technical facts, gathered from CloudDSF. The combination supports a benchmark and eventually a decision for an appropriate cloud vendor based on business-related and technical considerations.

2.5.3. Cloud Decision Support Framework Prototype

A prototypical implementation of CloudDSF, the Cloud Decision Support Framework Prototype (CloudDSF Prototype) was developed[7] [17]. The CloudDSF Prototype is a web application using standard web technologies, JavaScript Object No-

[7]The prototype is available at: `http://www.clouddsf.com`

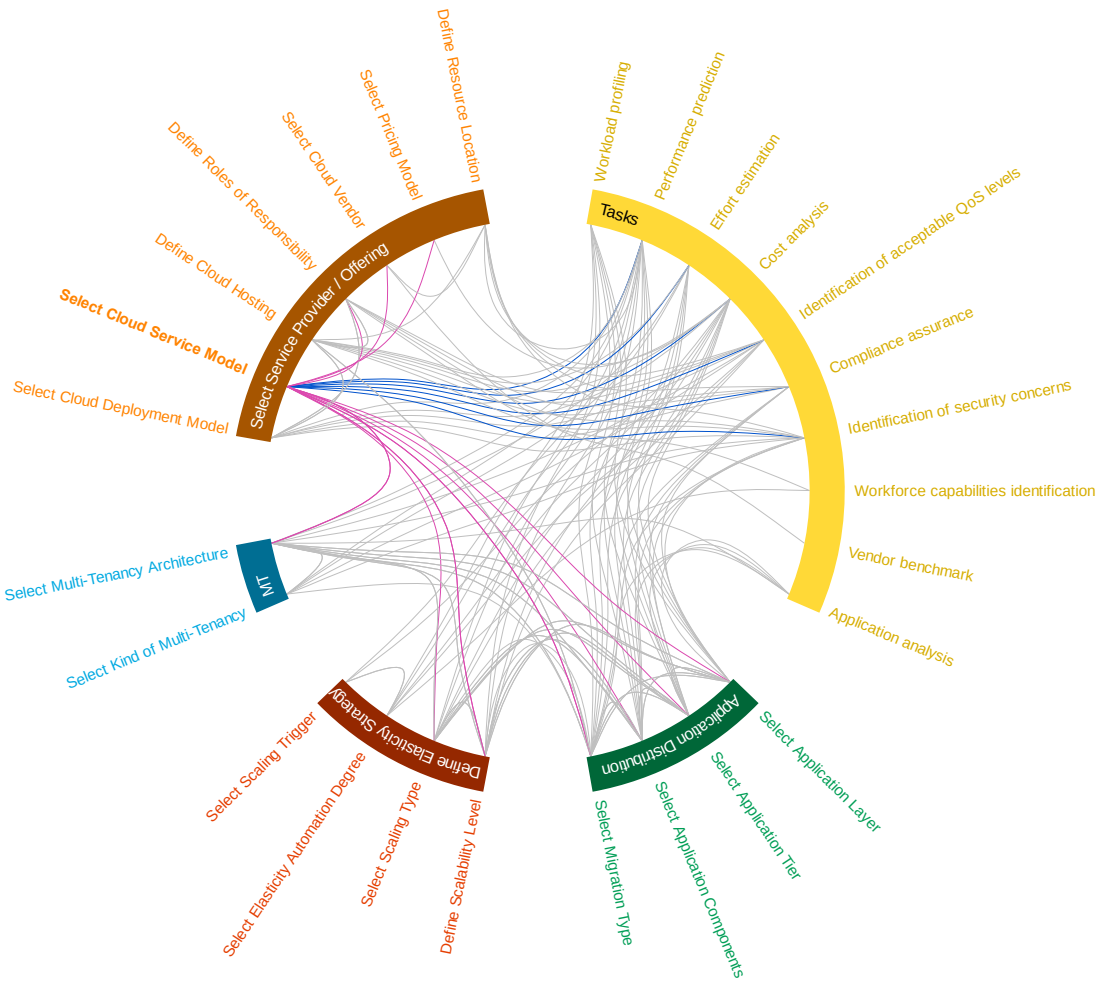

Figure 2.3.: Visualization of the relationships between tasks and decisions in the CloudDSF Prototype [46].

tation (JSON)[8] for encoding the CloudDSF knowledge base and the Data-Driven Documents (D3)[9] library as well as jQuery[10] for the visualizations. Its purpose is to provide decision makers with a platform-independent and publicly accessible implementation of CloudDSF to support them during the migration of applications to the cloud. An available visualization option is depicted in Fig. 2.3 showing the relations between decisions and decisions and tasks. Additional options are also available to visualize and to navigate through the CloudDSF knowledge base.

[8] http://json.org
[9] http://d3js.org
[10] http://jquery.com

2.5.4. Current Limitations of the Cloud Decision Support Framework

Several areas for further research to improve CloudDSF have been discussed in [18] and [17]. The elaboration and refinement of decision points conducted in [18] can be performed analogously for the tasks of CloudDSF, detailing their activities and their results relating to the different decisions. Also, a larger and more comprehensive evaluation than that which has been carried out should be conducted. The evaluation should include, besides the research, also the business domain to incorporate their requirements which might lead to further research topics and adaptations.

The current implementation in form of the CloudDSF Prototype enables a visualization of the different elements of CloudDSF and their relations, yet no actual interactive decision support. In order to provide a sophisticated DSS, the requirements for that desired system have to be defined and its corresponding components specified and implemented [18]. As a prerequisite, the relationships among decisions, tasks, between both of them and the connection to a given application model and the resulting necessary decisions have to be identified and formalized. Moreover, the relationships among outcomes and between outcomes and the inputs and outputs of tasks have to be elaborated.

The focus of this work is tackling the missing relationships between outcomes. As a prerequisite, a refinement of the existing knowledge base and, based on that refinement, a review of relations between decisions discovered in [18] will be conducted. Along the way, the suitability of the knowledge base for an identification of relations between outcomes will be addressed. This is followed by the previously stated elaboration and definition of the relationships between outcomes. Additionally, an enhancement of the CloudDSF Prototype to include and visualize the undertaken refinements and new relationships will be performed.

3. Refinement of the CloudDSF Knowledge Base

In this chapter the CloudDSF knowledge base will be refined and at the same time, the suitability of the outcomes for the identification of relations between them will be checked. Whenever necessary, decisions and outcomes will be adjusted. Additionally, the relationships among decisions as defined in [18] are reviewed and either confirmed, rejected, substituted or complemented by new relations based on the previously undertaken adjustments, see Section 3.5. In doing so, non-existing relations between decisions are verified as well. Therefore, the relations that have to be investigated on the more granular level between outcomes are reduced beforehand.

This chapter is divided into six sections. The first four correlate to the four decision points in which the decisions and their outcomes will be evaluated and refined as described above. In Section 3.5 the relations between decisions, within decision points and subsequently, the relations between decisions of different decision points are reviewed. The last section concludes the chapter with an updated knowledge base of the decision support framework, a summarization of the discovered relationship types between decisions and an quantification of the undertaken changes. For the rest of this work, entities of the CloudDSF knowledge base will be italicized to ease their identification.

3.1. Define Application Distribution

Originally named *Distribute Application* [16] but also referred to as *Application Distribution* [18], this decision point will be, for the sake of consistency, renamed to *Define Application Distribution*. The amount and types of decisions remain the same, however, outcomes under all decisions except for the decision *Select Migration Type* have been adjusted.

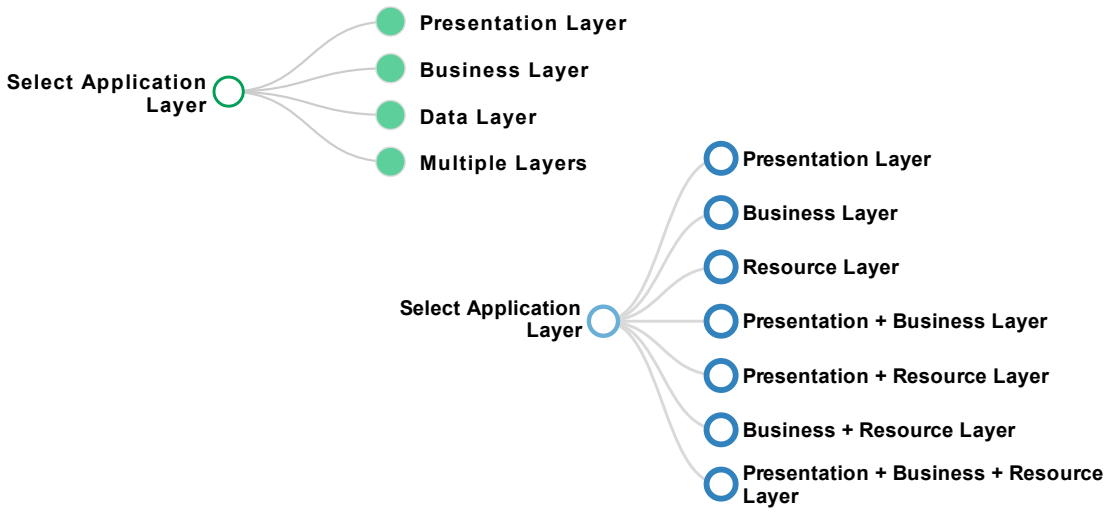

Figure 3.1.: Comparison between the previous and updated *Select Application Layer* decision.

Select Application Layer: In Fig. 3.1 the adjustments with respect to the decision *Select Application Layer* are depicted. In order to foster the distinctive character and avoid ambiguity between layers and tiers and therein their respective decisions, the outcome *Data Layer* has been renamed to *Resource Layer*. Moreover, the outcome *Multiple Layers* has been substituted by four outcomes denoting all possible combinations.

The underlying definition of the three application layers based on Fowler remains in place [47]. If a layer is chosen, the complete layer will be migrated with all its corresponding components whereat every layer naturally contains at least one component. While the presentation layer and business layer can contain application and middleware components, the resource layer is only comprised of the latter. During a migration, layers and their components might be reengineered, rewired to non-migrated layers, or wrapped into a virtual machine (VM), however, it is assumed that none of the components are moved to another layer.

Select Application Tier: Tiers denote the physical distribution of an application [47], [48]. Different mappings between layers and tiers are possible and also the amount of tiers that are actually used for a specific application. Besides the popular 3-tier architecture [48], [49], large web applications for instance, use multiple tiers with each tier solely dedicated either for web server, load balancer or databases for caching, in order to achieve a highly scalable and resilient application [50].

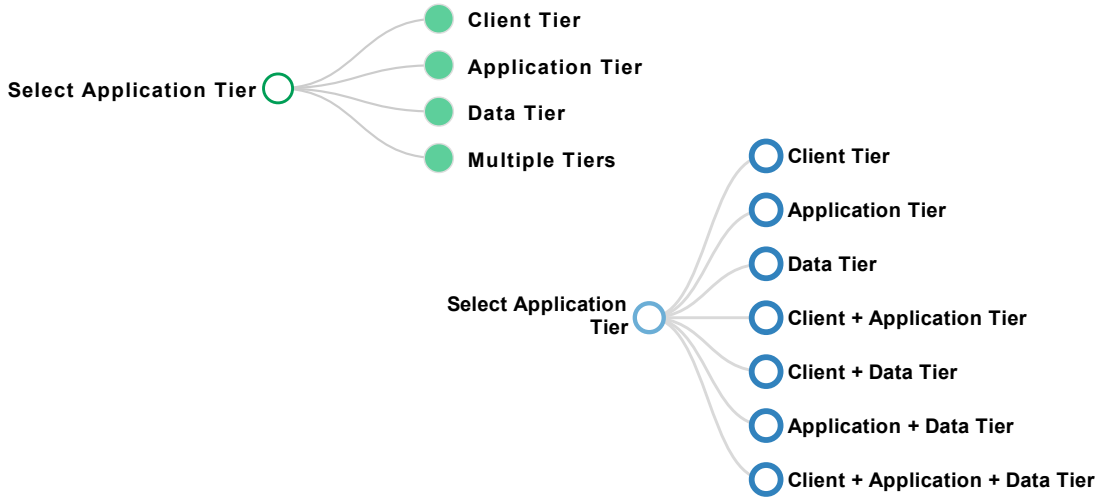

Figure 3.2.: Comparison between the previous and updated *Select Application Tier* decision.

As a result, without a specific application topology, mapping between layers and tiers and their relations cannot be inferred on a generic level. Logically, this holds true between components and tiers. Therefore, a tier selection always needs a refinement through a subsequent selection of components which equals a component selection in the first place. Therefore, no influencing relations between the decision *Select Application Tier* and other decisions can be stated, see Section 3.5.1. Hence, the decision has been detached from the other decisions and will not be further investigated in the following. Nevertheless, the decision will remain in the knowledge base to denote the necessary view on the physical distribution of an application prior to a migration as well as its significance for related tasks [18]. For the sake of consistency and completeness, the different selection possibilities have been added as outcomes as can be seen in Fig. 3.2.

Select Application Components: Significant changes have been undertaken with respect to the outcomes of the *Select Application Components* decision. The previous distinction between single and multiple components offered little additional information on top of a selection of layers or tiers. Even though a component should always and usually does belong to one layer [48], the obvious direct mapping between layer and component, as implied in [18], has been avoided. The separation of application functionalities and concerns into layers is merely a pattern and not a fixed requirement [47]. Thus, legacy applications might not follow this pattern or layers might have been intentionally merged [48]. In case only a single component

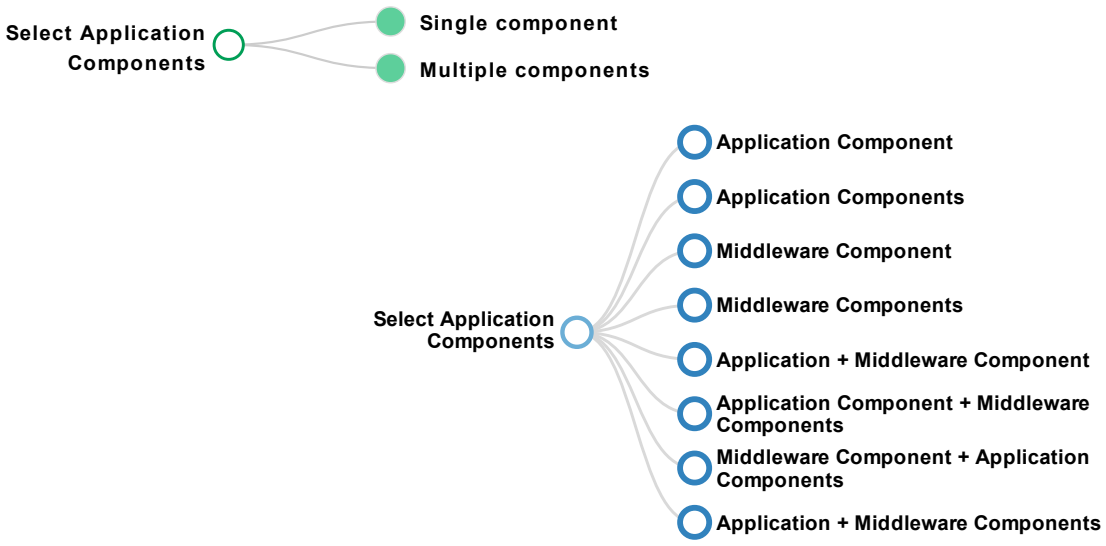

Figure 3.3.: Comparison between the previous and updated *Select Application Components* decision.

is migrated it is also important to know what kind of component it is and not only to which layer it belonged. To that end, two different types of components have been defined, see Fig. 3.3.

The first type and outcome *Application Component* denotes, e.g., an application front end war file and can be part of the presentation or business layer. The second outcome, *Middleware Component*, is denoting, e.g., a database management system, a message oriented middleware or an application server and can reside in any of the three layers. At first sight, middleware on the presentation layer seems counterintuitive, however, especially in a web application the presentation layer is very likely to contain middleware functionalities [51]. It must be pointed out that it is assumed that middleware components are only able to be migrated with an IaaS or PaaS solution whereas application components can be migrated with any service model. Due to the nature of middleware components and their specific use, a suitable SaaS can be considered impossible. Moreover, standard middleware components such as databases or application servers on their own are only offered as PaaS solutions since they do not deliver any form of application functionality per se, hence rendering themselves inept to be subsumed under SaaS [10].

The other five outcomes represent the multiple versions of each type and the resulting different combinations respectively. Thereby, the new outcomes denote not only the amount but also the type of the components that shall be migrated, thus enabling more detailed recommendations than before. Furthermore, relations on the

level of outcomes are now able to be identified which was previously not possible.

Select Migration Type: The four outcomes of the *Select Migration Type* decision remain the same but have been slightly renamed to provide self-describing outcomes. The new names and in order to avoid confusion, the assumptions and implications of a chosen migration type, based on [13], are as follows:

- **Migration Type I:** Replacement of one component by a cloud offering with any service model.
- **Migration Type II:** Partial migration of multiple components with any service model.
- **Migration Type III:** Migration of all layers, thus the whole software stack by wrapping it as-is into a VM and running it in an IaaS environment.
- **Migration Type IV:** Migration of all components, thus the whole software stack, solely based on cloud services with PaaS or SaaS for a higher leverage of cloud computing principles.

3.2. Define Elasticity Strategy

The decision *Define Scalability Level* has been significantly altered whereas decision *Select Scaling Type* remains untouched. The two remaining decisions *Select Elasticity Automation Degree* and *Select Scaling Trigger* have only been changed slightly.

Define Scalability Level: For the scalability levels, three main levels comprising five sublevels had been identified in [18]. This classification has been abandoned which is why the performed changes have affected all outcomes. The reasoning and development behind this result will be explained in the following.

The originally specified sublevels – virtual resource and physical hardware – are not in the scope of the cloud consumer who migrates the application. In any case, at least an IaaS solution would be used. Hence, the provider allocates virtual machines to the consumer who can only make requests towards the hypervisor or, more commonly, on a higher abstraction level to release or procure VMs and/or resources [10]. How the scaling is actually achieved below the VM level is decided and implemented by the provider. Even in an on-premise private cloud scenario, the decisions

regarding the migration of a legacy application would not be concerned with the internal setup of a private cloud and its scaling implementation. Consequently, these two outcomes have been removed.

Most application scaling strategies will eventually entail databases that often become a bottleneck in large web applications [52]. This scaling aspect has not been considered in [18]. Databases in general require a dedicated scaling strategy because, in contrast to application components, traditional relational databases are not stateless per se and therefore much harder to scale without making compromises regarding consistency, availability or reliability [52]. The efficient scaling of databases is an ongoing research topic, however, the most common approaches so far are the following:

- **Caching:** Dedicated databases store often accessed content to avoid read requests on the main database. Mechanisms are in place to ensure the currentness of the data [53].
- **Master/Slave Configuration:** One master database serves read and write requests whereas one or multiple synchronized slave databases only serve reads. Thus, the read load is distributed across several machines [54].
- **Master/Master Configuration:** Two or more master databases serving reads and writes. However, to ensure consistency between them the overhead often becomes so high that the performance is often decreasing instead of increasing [50].
- **Partitioning:** Splitting of tables within a database to reduce their size thus their index to enable faster access [54].
- **Sharding:** Horizontal partitioning of tables across different database server (shards). The corresponding shard for a request is calculated based on identifiers. The challenge is to find a good sharding configuration to evenly distribute read and writes that can be performed independently, i.e., without accessing multiple shards [52], [54].
- **NoSQL Databases:** Inherently scalable, distributed databases that are very often not suitable to replace databases of legacy applications since they rely on different assumptions regarding transaction handling and consistency [53].
- **Middleware:** Even though not directly a database scaling strategy, the middleware is often used to implement some of the aforementioned scaling strategies via redirecting or transforming requests [50], [53], [55], [56], [57].

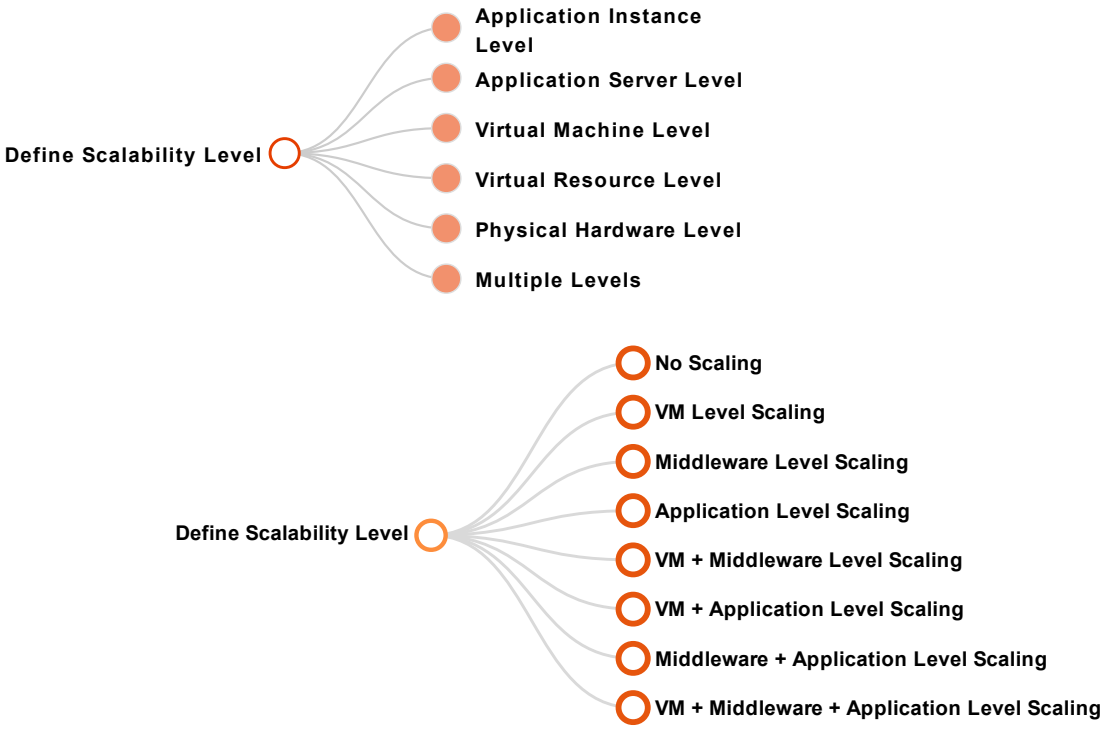

Figure 3.4.: Comparison between the previous and updated *Define Scalability Level* decision.

Several attempts to map these database scaling strategies to the remaining three scaling levels failed. Predominant cause was the non-existing hierarchical relationship between database scaling strategies as it is the case between VM, container and instance for the application components [53]. Databases are mostly limited by their underlying machines and their bare resources (e.g., I/O bandwidth, storage or memory). Scaling up a database therefore usually means setting up a complete new machine instead of just adding an additional database instance regardless of any logical connection as in the case of shards or slaves. In order to circumvent these problems while at the same time keeping the outcomes concise and simple, outcomes aligned to the possible migrated components have been chosen, see Fig. 3.4.

The outcome *VM Level Scaling* is the lowest possible level for scaling based on the aforementioned assumption that the migrator does not have control over the implementation of an IaaS solution. A one-to-one relation between operating system (OS) and VM is assumed. Recent container technologies such as *Docker*[1] are left

[1] `https://www.docker.com/`

out of the scope since they are not yet widespread and also of less interest for migrating legacy applications. The *Middleware Level Scaling* outcome subsumes all database scaling strategies as well as those scaling options previously subsumed under the container level such as scaling an application server by adding or removing instances. A third new outcome is *Application Level Scaling* comprising the scaling of application components such as an application instance hosted in an application server and corresponds to the previously highest scaling level.

In addition, a new outcome *No Scaling* has been added that naturally denotes the choice to not support any specific scaling. This outcome might be very reasonable to choose with respect to an application migration. For instance, an application with a very steady and well-known workload that is migrated as-is with *Migration Type III* to reduce the burden of infrastructure management would not benefit from scaling. Thus, the necessary reengineering effort would not be economically beneficial or necessary. Logically, the mentioned outcome cannot be part of any combination with other outcomes.

It is critical to mention that with respect to this decision it is assumed that the migrator wants to be in charge of the chosen scaling level. If, for example the outcome *Application Level Scaling* is selected, the migrator does not care how the underlying VMs or databases are scaled. Yet, he wants to be in charge of the number of running application instances and when and how they are scaled in or out. This assumption applies to all of the following scaling decisions.

Select Scaling Type: The *Select Scaling Type* decision has been reviewed and considered suitable as-is for the knowledge base and the refinement of relations on the level of outcomes. Hence, no changes have been carried out.

Select Elasticity Automation Degree: In addition to the original outcomes of this decision, two new outcomes denoting a third-party involvement, i.e., outsourcing the scaling operation, have been added, see Fig. 3.5. Outsourcing the *Manual Scaling* option does not seem appropriate since the scaling in this case is purely based on longtime single events and not performed on a day-to-day basis. The third party does not necessarily correspond to a used cloud vendor but rather to an independent organization offering scaling management on top of cloud vendor offerings such as *RightScale* [58]. Therefore, the assumption that the migrator wants to have control over what is scaled (*Define Scalability Level* and *Select Scaling Type*) in case of the involvement of a third party remains true. The *Select Scaling Trigger* decision on the other hand is subject to the third party, see Section 4.2.3 for further details.

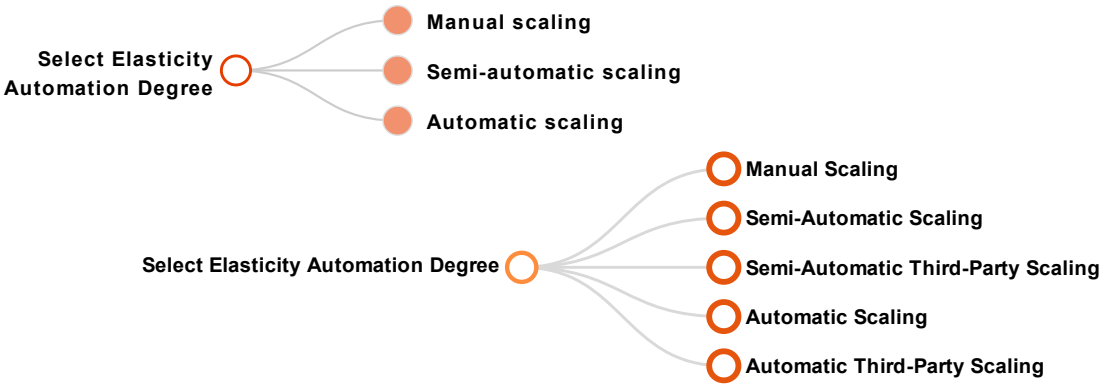

Figure 3.5.: Comparison between the previous and updated *Select Elasticity Automation Degree* decision.

Select Scaling Trigger: The new outcome *No Trigger* has been added to the *Select Scaling Trigger* decision to offer a corresponding option for the *Manual Scaling* outcome of the *Select Elasticity Automation Degree* decision. Even though a user might get a notification in case of a manual scaling approach the notification would not "trigger" any action directly. The new outcome and, for the sake of consistency, the renaming of the other two can be obtained from Fig. 3.6.

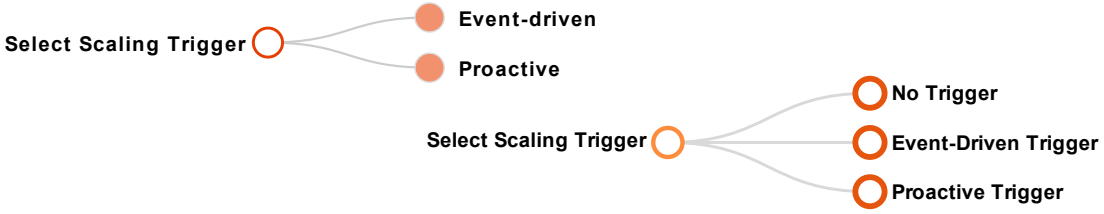

Figure 3.6.: Comparison between the previous and updated *Select Scaling Trigger* decision.

3.3. Define Multi-Tenancy Requirements

This decision point is the only one where a complete decision has been removed. The two outcomes of the decision *Select Kind of Multi-Tenancy* had been, as intended, encoded in more detail in the decision *Select Multi-Tenancy Architecture* [18]. For example, multi-tenancy on the level of the application instance and database instance or schema respectively were equal to the outcome *Native Multi-Tenancy* [59], [60].

Consequently, the decision has been discarded to avoid confusion and redundancy within the knowledge base.

In a first attempt to reduce the number of multi-tenant architectures (MTAs) a new matrix was created omitting the hardware and virtual machine level while including the operating system level leading to a total of 19 MTAs [59]. However, despite the still very high number of outcomes several problems inherent to the classification in MTAs, that are described in the following, remained, inhibiting an intuitive understanding and rendering them inappropriate to actually support decision makers. A MTA is very detailed and entails an independent view of the application and resource layer leading to problems if only parts of the application are migrated. Furthermore, the adjacent number to a MTA does not necessarily indicate a higher or lower level of multi-tenancy – the amount of sharing between tenants – thus further obscuring their meaning. Also, relationships between scaling, components and the MTAs were hard to infer. Thus, another approach to define multi-tenancy was taken abandoning MTAs completely.

In literature and practice, multi-tenancy is divided up into three to six levels [60], [61], [62], [63], [64]. Even though there are different opinions at what point one can speak from multi-tenancy [65], for this work, the definition from Andrikopoulos et al. is used defining multi-tenancy as "the sharing of the whole technological stack (hardware, operating system, middleware, and application instances) at the same time by different tenants and their corresponding users" [13, pp. 25-26]. A completely dedicated technological stack equaling a normal application service provider without any sharing is therefore not in the scope of this decision even though it can be argued that, from the consumer point of view, some flavor of multi-tenancy exists [62]. Ultimately, the goals and desired benefits of multi-tenancy are a better resource utilization, a shared code base, a common data structure and the possibility to customize the application on a per tenant basis [62], [65], [66].

The different levels for sharing the database among tenants are as follows: at the database server level, at the database management system level (i.e. dedicated database instances), at the database instance level (i.e. dedicated tables) or at the schema level (i.e. dedicated rows) [64]. The first two levels are very similar regarding data isolation and engineering effort which is why, due to the better resource utilization, the second is considered preferable. The latter two have both a higher implementation complexity and less secure data isolation [67]. Those previously from the application view separately defined database options are now, without actually sacrificing much expressiveness, combined in a more coarse grained classification with four multi-tenancy levels (MTLs). Each outcome of the decision corresponds to one MTL as visualized in Fig. 3.7 and described in the following:

- **Shared Hardware Multi-Tenancy (MTL 0):** The lowest level of multi-tenancy shares the same hardware and physical resources. On this level every tenant has its own VMs and the level of separation is very high. However, since cloud consumers have no influence on how multi-tenancy is implemented on the hypervisor level, the provider has to ensure the logical separation between the tenants' VMs to avoid performance or security problems [68], [69].
- **Shared OS Multi-Tenancy (MTL 1):** On the next higher level tenants share the OS and, through an assumed one-to-one relation, also the underlying VM. The databases of the tenants will still be completely separated either through a dedicated database server or through a dedicated instance within a shared database server avoiding any extensive adaptations to the data structure [64].
- **Shared Middleware Multi-Tenancy (MTL 2):** The third level shares the middleware among tenants. This can include, for example, the database or the application server, but not the application instance. The database is normally shared on the database server level or on the instance level [61]. However, theoretically any multi-tenancy approach for the database as described above can be applied [62].
- **Shared Application Multi-Tenancy (MTL 3):** The highest multi-tenancy level, often described as native multi-tenancy or single-instance multi-tenancy, shares the actual application instance among tenants. This approach is especially popular among SaaS providers with several hundreds to thousands of tenants [60], [62]. In those cases the database is usually shared on a table or schema basis enabling the best resource utilization, a common code base and data structure while still enabling extensive customization.

The new outcomes form a continuum of increased resource utilization but also reengineering effort and decreasing isolation as depicted in Fig. 3.7. Thereby, an outcome defines only a minimum of shared resources but does not dictate a detailed implementation, i.e., if the database has to be shared on instance or schema level in the case of sharing the middleware. It is critical to mention that multi-tenancy is now an application wide decision. Hence, if a MTL is chosen, the whole application has to be migrated and has to support multi-tenancy according to the specified level.

In order to ensure the separation of tenants, only certain service models can be used in combination with a MTL, see Fig. 3.7. The responsibility for the desired level of separation between tenants must be on the consumer side because it is impossible to guarantee a separation through the cloud vendor through the obfuscation of their architecture and implementation [13]. For instance, *Shared Middleware Multi-Tenancy*

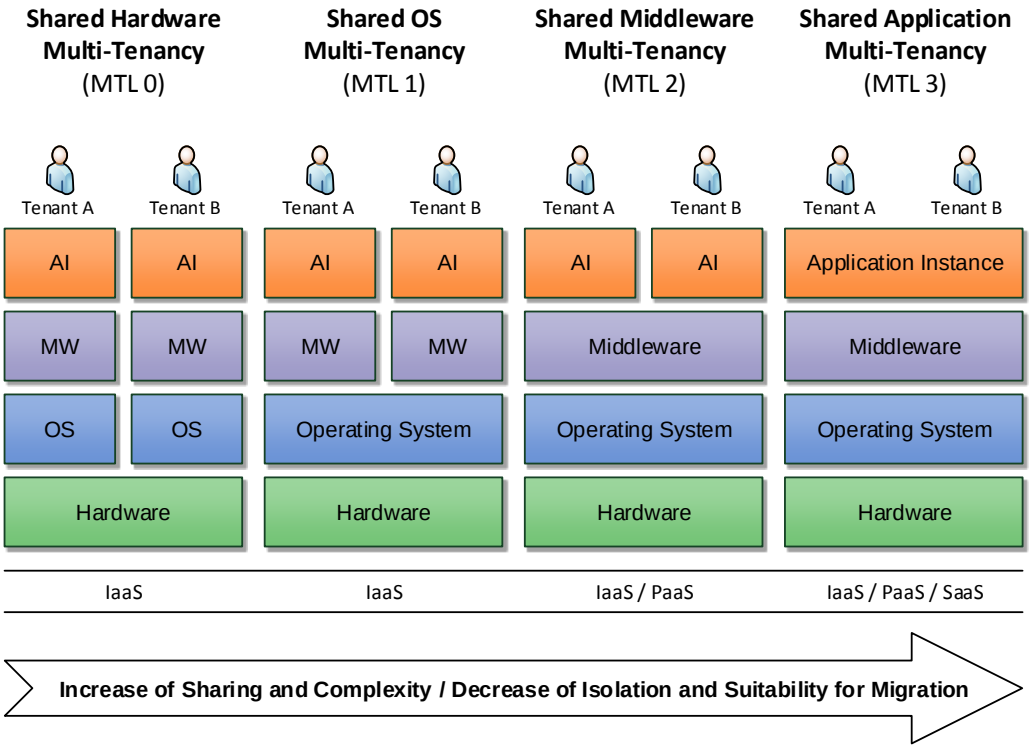

Figure 3.7.: The four multi-tenancy levels and their applicable service models.

cannot be implemented with SaaS because that service model already implies a possible sharing of the application instance [10]. Due to the new classification in levels, the decision has been renamed from *Select Multi-Tenancy Architecture* to *Select Multi-Tenancy Level*. The new structure of the whole decision point in comparison to before the refinement can be obtained from Fig. 3.8.

In essence, the main alternative to choose from with respect to application migration is either between *Shared Hardware Multi-Tenancy* and *Shared OS Multi-Tenancy* or *Shared Middleware Multi-Tenancy* and *Shared Application Multi-Tenancy* because the former two enable the recycling of most precloud application components (e.g., legacy middleware is often not multi-tenant aware), making it suitable for migrations, while the latter two require a newly designed container and or new programming model [62].

Since the multi-tenancy approach for the data is more relevant to the performance than the application level [70], most application providers choose a dedicated database for each tenant. Often, due to the high workload of tenants, dedicated databases are needed anyway [64], [66]. Also, the separation of the tenants' sen-

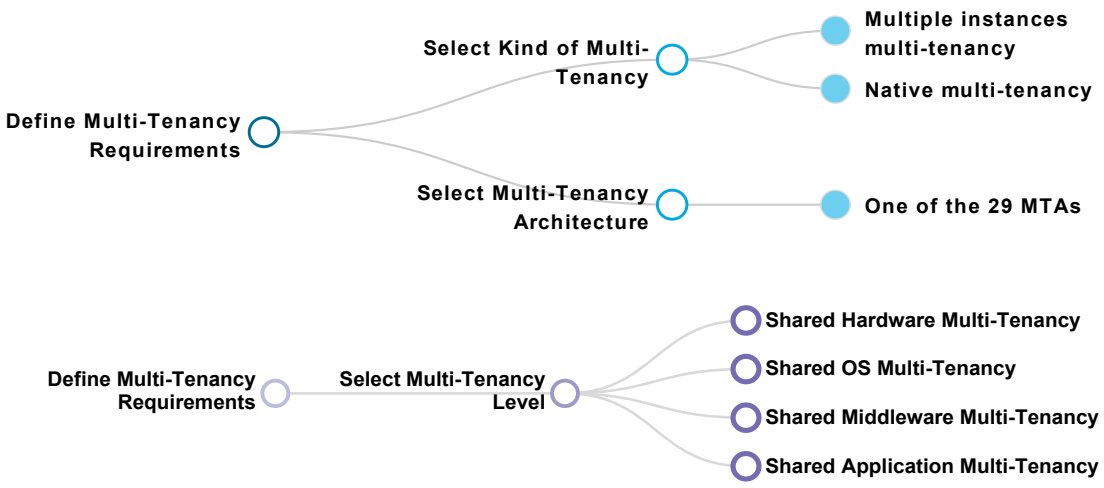

Figure 3.8.: Comparison between the previous and updated decision point *Define Multi-Tenancy Requirements*.

sitive data is easier to ensure. *Microsoft*, for example, migrated their enterprise resource planning system to the cloud. They share the application server among tenants but provide each one of them with a database instance that share the database server and the underlying hardware [71]. As already stated, *Shared Application Multi-Tenancy* is very costly and time-consuming for migration projects and is therefore predominantly used by native SaaS providers such as *salesforce* [66], [72]. For new cloud applications that require multi-tenancy for a large amount of tenants though, it can be considered as the preferred model [62].

3.4. Select Service Provider / Offering

Every decision except the *Select Cloud Vendor* decisions has undergone changes, however, only the decisions *Define Roles of Responsibility* and *Define Resource Location* were significantly altered.

Select Cloud Deployment Model: The outcomes of the *Select Cloud Deployment Model* decision have not been modified. Additional outcomes denoting all possible combinations are not necessary due to the similarities of private and community clouds regarding risks of multi-tenancy, security or performance and resource limitations [10]. Thus, the outcome *Hybrid Cloud* will be defined as the combination of the outcomes of *Public Cloud* and *Private Cloud* and/or *Community Cloud* respectively.

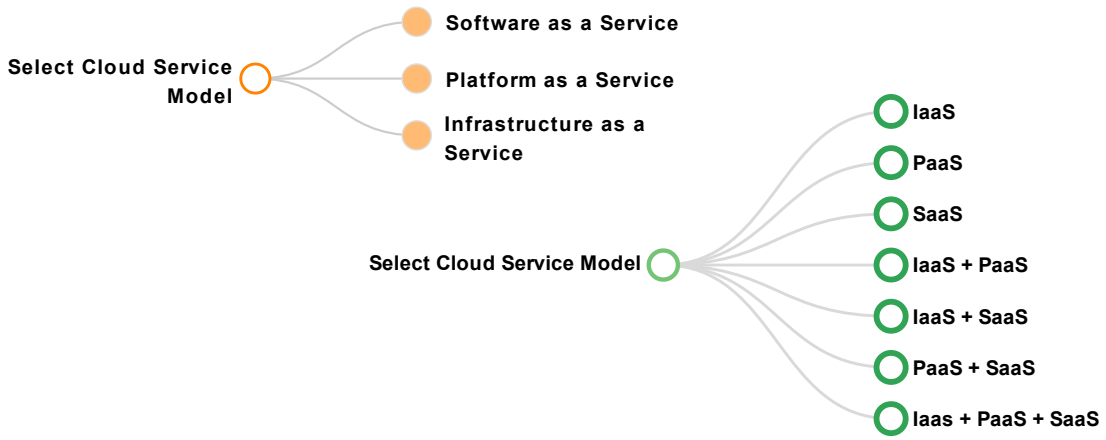

Figure 3.9.: Comparison between the previous and updated *Select Cloud Service Model* decision.

Select Cloud Service Model: The outcomes of this decision have been complemented by their different combination possibilities as can be seen in Fig. 3.9. Due to their ubiquitous use and well-known meaning, the abbreviations of the three respective service models: infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS) have been chosen as names for the outcomes to keep them short and manageable, especially for the combinatorial outcomes.

Define Cloud Hosting: The outcomes of this decision have been, for the sake of consistency and ease of use, slightly renamed omitting the term “cloud”. The new names are *On-Premise Hosting*, *Off-Premise Hosting* and *Hybrid Hosting*. Any other adaptations have not been necessary.

Define Roles of Responsibility: The amount of possible roles have been reduced from eight to three. For clarification and a better understanding why the roles have been altered, the three different categories defining the roles are described in the following [10], [73]:

- **Ownership:** The ownership role defines who owns the physical hardware, e.g., server, storage, network hardware and so on. In accordance with cloud computing benefits it can be expected that a migration should also increase flexibility with regard to the possessed infrastructure. Therefore, it can be assumed that in case of an off-premise hosting the ownership belongs to a third party. Naturally, this applies anyway in case of a public cloud.

- **Operation:** The operation role subsumes activities regarding infrastructure maintenance, incident management, system and information integrity, protection and so on. The amount and variety of the activities change depending on the service model. For instance, if a cloud provider offers PaaS he has to take care amongst other things of the aforementioned activities plus scaling, multi-tenancy and update for the platform level and the levels below.
- **Management:** The management role comprises all activities that would normally be performed by a cloud consumer regarding the specific service model. In the case of SaaS, possible responsibilities would be account management and configuration of the procured application within the limits of the provided options. IaaS would add application deployment, management of virtual machines, operating systems and backups and many more tasks to the list.

Naturally, the chain of infrastructure outsourcing starts with the lower level tasks. It seems therefore highly unlikely that as described through *Role Set 3* in [18], a cloud consumer would operate the cloud environment but not manage it. The same applies to *Role Set 2* and *Role Set 4*. Furthermore, the trend for on-premise clouds is going towards buying complete out-of-the-box solutions [74]. That means that hardware and software is already efficiently bundled into big racks providing the customer with a pre-configured, easy to use private cloud platform, thus minimizing and facilitating the operational tasks in the first place. Hence, *Role Set 2 – Role Set 4* have been discarded.

Even though there are offers installing third-party owned hardware on-premise that can be operated by the consumer or not [75], one of the predominant reasons for a cloud migration is the freeing of the infrastructure burden, the flexibility and the outsourcing of the risk of under- or over-provisioning [5], [23], [24]. Also, buying hardware is, in the long run, normally monetarily better for the consumer than leasing it. Given high workloads, the aforementioned out-of-the-box solutions would be favorable. Naturally, taking the burden of operating the cloud environment for third-party owned hardware seems unlikely and in case of a public cloud or off-premise hosting it is out of the question anyway. Hence, *Role Set 5* and *Role Set 6* can be considered negligible and have been removed.

The performed changes, the three remaining outcomes and their possible combinations can be seen in Fig. 3.10. For better comprehensibility and indication of their meaning, the names of the outcomes have been changed. In case of *Role Set 1* everything is controlled by the company migrating the application, hence the outcome has been renamed to *Inhouse*. *Role Set 8* is the exact opposite, thus the outcome has been named *Outsourced*. *Role Set 7* leaves the management part to the company hence the name *Management*.

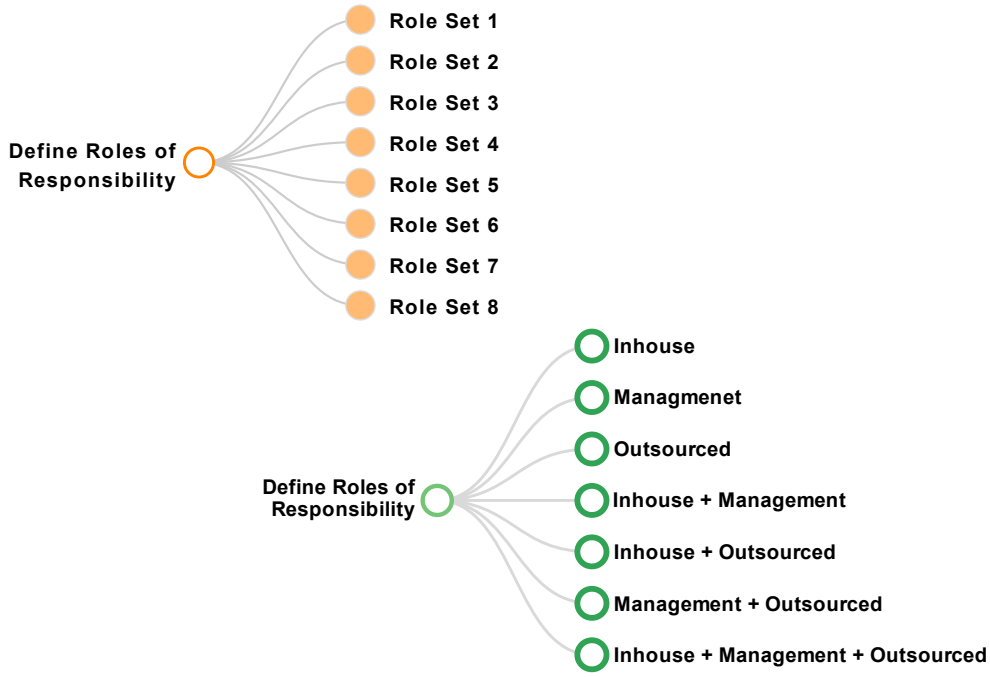

Figure 3.10.: Comparison between the previous and updated *Define Roles of Responsibility* decision.

Select Cloud Vendor: The cloud vendor outcome is currently defined as the result of an evaluation of cloud vendors through the task *Vendor Benchmark* which considers primarily "soft facts" [18]. In order to select a cloud vendor based on the selected technical outcomes in CloudDSF, it would be necessary to thoroughly examine various vendors regarding their provided features according to the decisions, i.e., their offered scalability options, service models, etc. and create the respective outcomes for each property.

The vast amount of cloud vendors as well as the plethora of different offerings would exceed the scope of this thesis. For instance, *Amazon* as the biggest, yet only as one cloud vendor, already lists 53[2] different cloud products and services with highly variable pricing options and combinations. As a consequence, the outcome will not be further refined. Instead, two new relationship types will be introduced to reflect the dependencies of outcomes towards the cloud vendor, see Section 3.5 for further details.

[2]The number of services has been derived based on the listed products at `http://aws.amazon.com/products/` (date of access 12/25/2014).

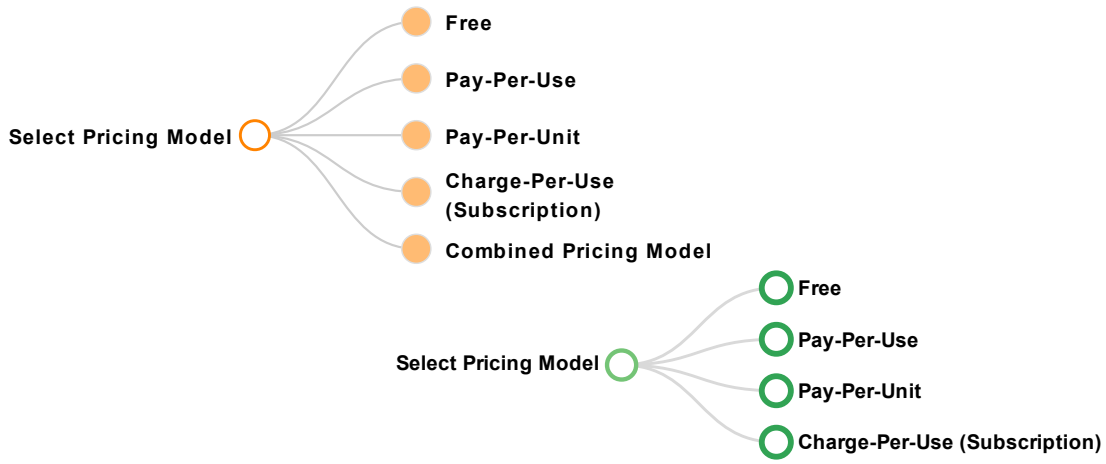

Figure 3.11.: Comparison between the previous and updated *Select Pricing Model* decision.

Select Pricing Model: Similar to the previous *Select Cloud Vendor* decision, it is not feasible to examine all offered pricing options and combinations of different vendors. For example, in a recent survey more than 16.000 prices have been tracked for only six cloud vendors [3]. Also, a migrated application might consist of multiple services, thus tremendously increasing the pricing options and combinations especially with regard to elasticity [22], [76]. Yet, the outcome *Combined Pricing Model* on the other hand would be too unspecific to infer any relations and would contradict the pattern of explicitly stating all possible combinations among outcomes. As a result, the outcome was removed resulting in a total of four outcomes as can be seen in Fig. 3.11, intentionally simplifying the decision.

Define Resource Location: The original outcome lacked expressiveness for inferring relations and has been replaced by two new outcomes indicating where the resources, i.e., the application data, have to be kept. The data location is of crucial importance because depending on the geographical location, data, regardless of its origin or usage, might fall under the local laws and regulations [77], [78]. Specific regions or countries as outcomes have been deemed inappropriate since a company can operate in various regions and coverage of all possibilities would not be possible. Besides, the country is of less importance but rather the corresponding data regulation laws. The high amount of regulations often depending on the business domain, the uncertainty of their application and their outdated national character thus inappropriateness for cloud computing commonly involving data transfers across borders, renders them unsuitable as outcomes as well [77], [79], [80].

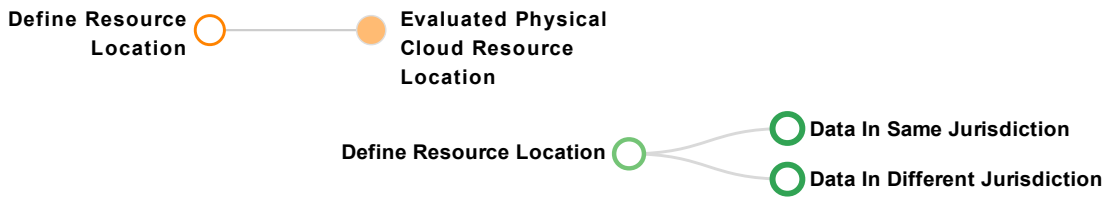

Figure 3.12.: Comparison between the previous and updated *Define Resource Location* decision.

Instead, a higher level – the jurisdiction of the company – has been used to define the outcomes, visualized in Fig. 3.12. They denote whether the data has to be stored within the same jurisdiction that the company operates in or not. This approach therefore does not indicate specific regulations or laws but rather rise the awareness if the migration is bound by data regulations or not and how that relates to other decisions. An elaboration of whether a different jurisdiction can be used and which specific laws and regulations will actually apply must be performed as part of the task *Compliance Assurance* as a preceding and/or subsequent step, respectively. Other considerations in relation to geographical distribution such as performance, latency or cost were not and have not been included into the scope of this decision.

3.5. Review of Relations Between Decisions

In the following, all relations between decisions defined in [18] will be reviewed and either confirmed, substituted or removed. To keep the actual review concise the three newly identified relationship types will be briefly summarized and described beforehand to foster a common understanding. The *Influencing* relationship type defined in [18] remains untouched, indicating that a selection of an outcome influences the possible selectable outcomes in the related decision. Influencing relations that remain valid under the new assumptions and outcomes defined in Chapter 3, will briefly be confirmed only. Further details regarding their influence can be obtained from Chapter 4.

As explained in Section 3.4, the *Select Cloud Vendor* decision has not been refined. In order to denote relations, two new relationship types have been introduced. Relations from the *Select Cloud Vendor* decision are denoted by *Binding* relations. That means that a decision and its connected outcomes are subject to the cloud vendor regarding support or exact implementation. The other way around, relations towards the *Select Cloud Vendor* decision are denoted as *Affecting* relations. That means that

the decision is imposing certain requirements towards a cloud vendor. Therefore, those decisions affect the selection of an appropriate cloud vendor.

Several decisions require a subsequent selection of another decision. For instance, as stated in Section 3.1, it cannot be inferred what kind of components or layers reside on a tier. Therefore, the *Select Application Tier* decision requires the *Select Application Components* decision for further refinement. These relations are denoted by a new *Requiring* relationship type. It must be mentioned that in some cases a decision might require more than one decision. If, however, decision A requires decision B and decision B requires decision C, a requiring relation between A and C can be inferred (transitivity) and will be encoded through the two respective requiring relations.

In regard to influencing relationships, a different approach applies. If decision A influences decision B and decision B influences decision C, an influencing decision between decision A and C cannot be inferred, differing from the approach applied in [18]. In other words an influencing relationship between two decisions cannot be argued via the influence over another decision. However, if decision A influences decision B that influences decision C and decision A influences decision C independently from decision B, of course, a relation will be stated.

All relations defined in [18] are directed, thus there is always a source and a target decision. An influencing relation from a decision A towards a decision B does not imply an inverse influencing relationship. This definition will be kept and applied to *all* relationship types. A summary of the relationship types, including an example, can be found in Section 3.6.

3.5.1. Relations Within Decision Points

In order to keep the review comprehensible, first the relations between decisions within decision points are examined. Subsequently in Section 3.5.2 the relations between the decisions of different decision points are reviewed. Wherever appropriate, the term “decision” is omitted to keep the review less verbose, though the decisions can be easily identified through the applied italicization. A detailed listing of all identified decision relations which are described below can be found in the appendix in Table A.1 and Table A.2.

Relations within Define Application Distribution: All influencing relations between *Select Application Tier* and other decisions cannot be confirmed and have been removed, see Fig. 3.13. As stated in Section 3.1, concrete mappings between layers

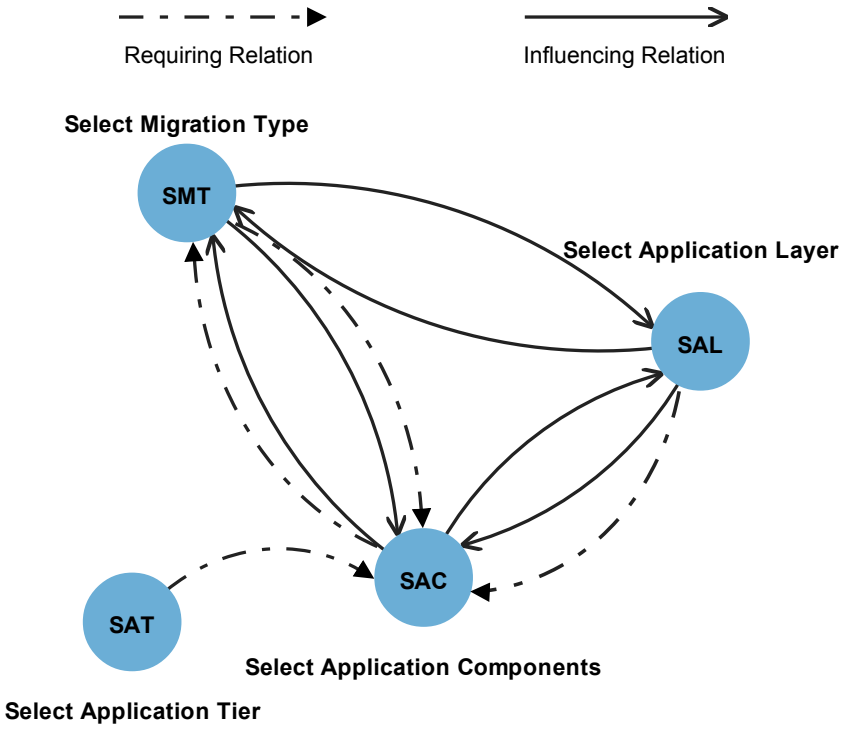

Figure 3.13.: Decision relations within *Define Application Distribution*.

and tiers or tiers and components cannot be inferred. The same applies towards *Select Migration Type* since a selection of one tier does not indicate if the complete application or just part of it is migrated.

The influencing relations from *Select Application Layer* and *Select Application Components* towards *Select Migration Type* and vice versa can be confirmed. The determining relationship between *Select Application Layer* and *Select Application Components* has been substituted by a normal influencing relation since the assumptions and outcomes of the latter decision have changed. The reverse influencing relationship can be confirmed and still holds true under the new circumstances.

To enable a more detailed recommendation a selection of the migrated components is necessary. Therefore, all decisions have a new requiring relation towards *Select Application Components*. That means for example, in the case a tier is selected a subsequent selection of components is required. In reverse, one requiring relation towards *Select Migration Type* has been identified. If multiple components are selected it is not yet clear if they comprise the whole application which would exclude *Migration Type II* or if they represent a partial functionality.

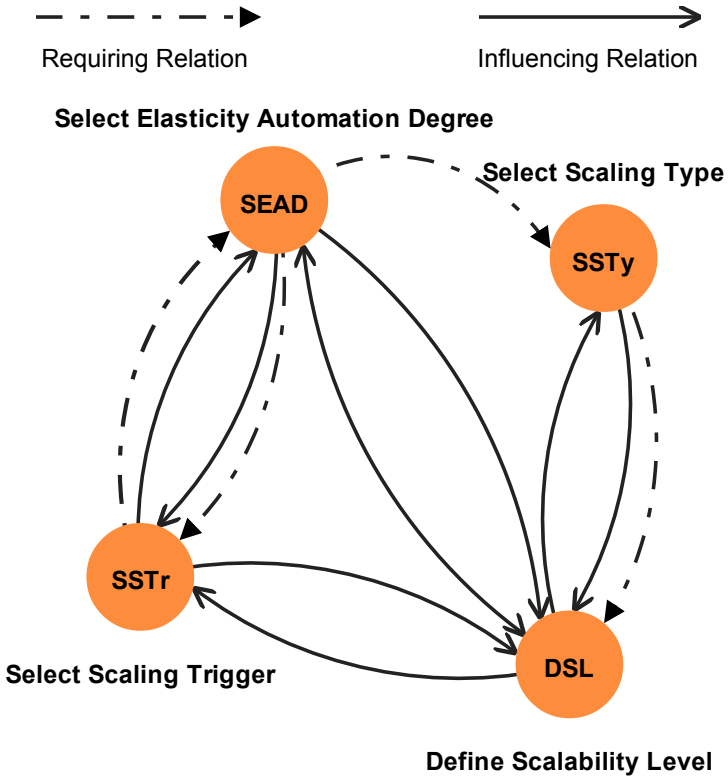

Figure 3.14.: Decision relations within *Define Elasticity Strategy*.

Relations within Define Elasticity Strategy: All defined relationships in [18] can be confirmed. Additionally, the decision *Define Scalability Level* has, due to its new outcome *No Scaling*, two new influencing relations towards *Select Elasticity Automation Degree* and *Select Scaling Trigger* since the outcome, if selected, will render a trigger or automation useless. Furthermore, four new requiring relations have been identified, see Fig. 3.14. One from *Select Elasticity Automation Degree* towards *Select Scaling Trigger* and vice versa. Either decision cannot be chosen on its own. Just selecting a trigger outcome does not make sense because some sort of automation degree is necessary. This applies analogously in reverse.

Another requiring relation exists between *Select Scaling Type* and *Define Scalability Level*. The type of scaling is directly dependent on the scalability level, hence if a scaling type is chosen a certain level has to be selected as well. The last requiring relation exists between *Select Elasticity Automation Degree* and *Select Scaling Type*. Any automation needs a certain scaling type otherwise there would be nothing to automate. Logically this applies towards the level as well. However, through the aforementioned requiring relation between the scaling type and scaling level, the

level will be, through the chain of required relations, selected after all. Therefore, additional requiring relations are not necessary and are already encoded in the transitive nature of the relations.

Relations within Define Multi-Tenancy Requirements: As an immediate result of the conducted adaptations in Section 3.3 only one decision under the decision point remains, rendering all relations within this decision point obsolete. Naturally they will not be reviewed and have been removed.

Relations within Select Service Provider / Offering: All influencing relations can be confirmed except those included in the *Select Cloud Service Model* decision. In contrast to the statement in [18], a service model does not influence or determine a certain pricing model and vice versa. Indeed, specific combinations of pricing and service models are popular and more likely, such as IaaS and pay-per-unit or pay-per-use. However, there are too many varieties and combinations in such pricing, complicating a possible generalization [75], [76], [81], [82]. Furthermore, the responsibilities are distributed to some extent regardless of the service model [73], [83], see also Section 3.4

Several new relations have been identified. One new influencing relationship exists between *Define Resource Location* and *Define Cloud Hosting*. For instance, if *Data In Different Jurisdiction* is chosen, the option *On-Premise Hosting* is no longer allowed because naturally, the data would be stored in the same jurisdiction instead of a different one. Furthermore, all decisions within the decision point have an affecting relation towards *Select Cloud Vendor* and a binding one in reverse, see Section 3.5. Two previously existing influencing relations have been substituted accordingly. The reasoning is very straightforward. As soon as a cloud vendor is involved, which could be the case in any of the decisions or if the decision actually requires it, for example in case of a public cloud, the vendor has to support the desired functionalities or in reverse, restrict the available possibilities.

In addition, multiple requiring relations have been identified, see Fig. 3.15. The *Select Cloud Deployment Model* decision has one towards *Define Cloud Hosting* and one towards *Select Cloud Vendor*. The combination of the deployment model and the hosting option is essential. One cannot go without the other since the implications, requirements and properties vary significantly between, for example, an off-premise or an on-premise private cloud [10]. Also, a public cloud indeed requires off-premise hosting. The same applies for the cloud vendor. In the case of a public cloud, a vendor has to be selected that actually serves as the cloud provider. Another

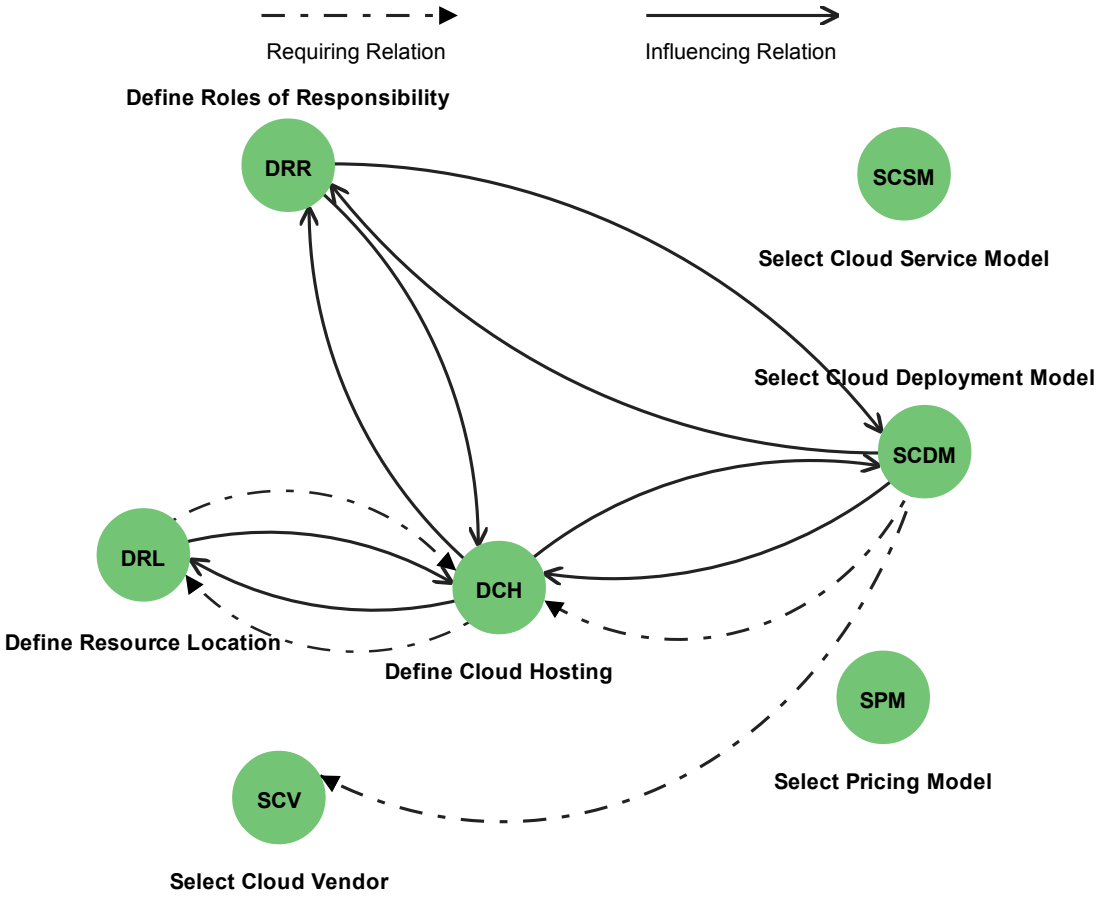

Figure 3.15.: Decision relations within *Select Service Provider / Offering*.

requiring relation exists from *Define Cloud Hosting* towards *Define Resource Location* and vice versa. In order to raise the awareness of the data location in case of an off-premise hosting, the resource location has been deemed necessary to be defined. The same holds true in reverse.

3.5.2. Relations Between Decision Points

Following the same order used previously, in this section all relations of the decisions of one decision point towards other decision points' decisions will be revised. Relations towards and from the removed *Select Kind of Multi-Tenancy* decision are obsolete and their removal will not be explicitly mentioned in the following. For reference, within every paragraph a short figure depicting the identified influencing and requiring relations for the respective decision points towards others will be pro-

vided. A summary and visualization of the other relationship types can be found in Section 3.6.

Relations from Define Application Distribution towards other decision points: Towards the decision point *Define Elasticity Strategy* all four influencing relations from *Select Application Layer* and *Select Application Tier* have been deleted detaching the decisions from any influence, see Fig. 3.16. As previously mentioned, a tier does not indicate which components are migrated or if the tier comprises the whole or only part of the application. Both, layer and tier neither have had any influence towards the *Define Scalability Level* or *Select Scaling Type* decision. They are too unspecific to indicate a necessary scaling on any level. Furthermore, the scaling type is influenced and dependent on the level, hence only indirectly influenced by the aforementioned decisions, as correctly stated in [18]. In that case, an indirect influence via *Define Scalability Level* is sufficient and both relations have been removed. The same also applies for *Select Application Components* or *Select Migration Type* whose relations towards *Select Scaling Type* have been removed as well.

Relating to the decision point *Define Multi-Tenancy Requirements* the relation between *Select Application Tier* and *Select Multi-Tenancy Level* has been deleted. A tier does not indicate if the complete application has been chosen for migration which is a prerequisite for enabling multi-tenancy, see Section 3.3. Therefore, an influence cannot be confirmed.

With respect to decision point *Select Service Provider / Offering*, three relations have been removed. The first two existed from *Select Application Layer* and *Select Application Tier* towards *Select Cloud Service Model*. In [18] it is argued that a single application component, although at that time not yet defined as outcome under the *Select Application Components* decision, would be likely migrated with PaaS or SaaS. Firstly, a layer or tier does not exactly indicate what kind of components are migrated. Secondly, a preferable option is not equal to a real influence. Therefore both relations have been removed. The third removed relation existed between *Select Migration Type* and *Define Cloud Hosting* denoting that a certain migration type would restrict the hosting options [18]. In fact, Andrikopoulos et al. specifically consider any form of hosting possibility regardless of the migration type and only distinguish between traditional (non-cloud) and cloud resources [13].

One new requiring relation has been identified from *Select Application Components* towards *Select Cloud Service Model*. Through the chain of requiring relations within the *Define Application Distribution* decision point, ultimately, the application components and the service model will be chosen no matter what. This leads to a necessary

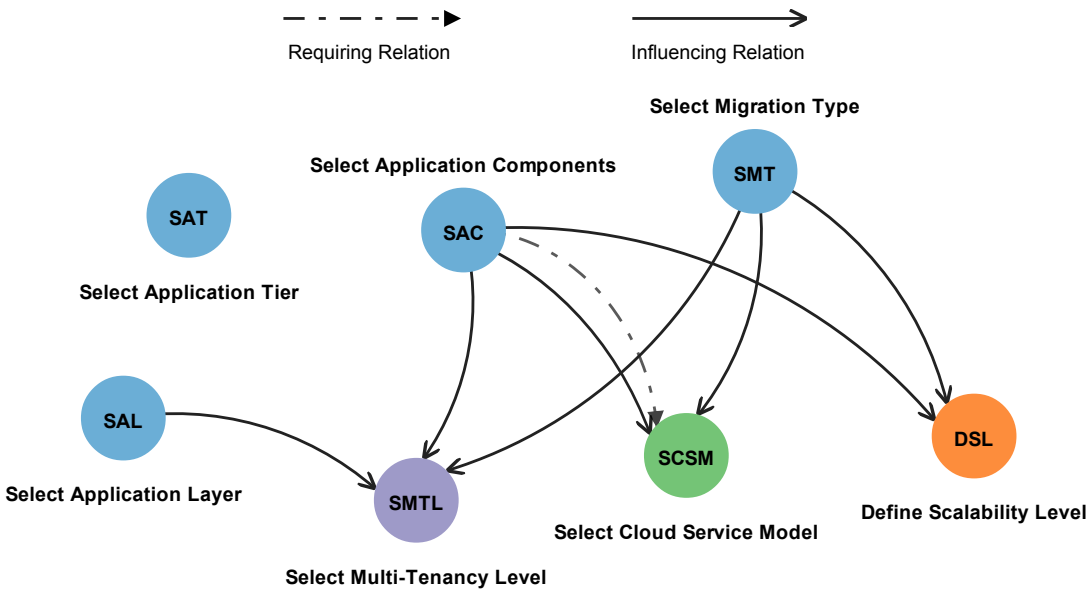

Figure 3.16.: Outgoing decision relations of *Define Application Distribution*.

minimum of answered decisions in order to be able to give recommendations to the decision maker. All other remaining relations between this decision point and others can be confirmed.

Relations from Define Elasticity Strategy towards other decision points: For the relations from *Select Scaling Type* towards all decisions of the *Define Application Distribution* and *Define Multi-Tenancy Requirements* decision points, the same argument as previously described for the reverse relations applies. Therefore five relations cannot be confirmed and have been removed. The same is valid for the two relations from *Define Scalability Level* towards *Select Application Layer* and *Select Application Tier* respectively that have been removed as well. This leads to only three remaining relations towards those decision points as can be seen in Fig. 3.17.

One new influencing relationship has been identified between *Define Scalability Level* and *Select Migration Type*. A selection of the *No Scaling* outcome for example influences the available migration types because it is defined that *Migration Type IV* fully leverages cloud computing capabilities, e.g., based on SaaS including scalability [10] with the goal of a cloud-enabled application [13]. Therefore some kind of scaling is necessary. Besides, *Migration Type I* depicts the migration of one component thus inhibiting combinatorial scaling levels that would require different components. Therefore, an influencing relationship is appropriate.

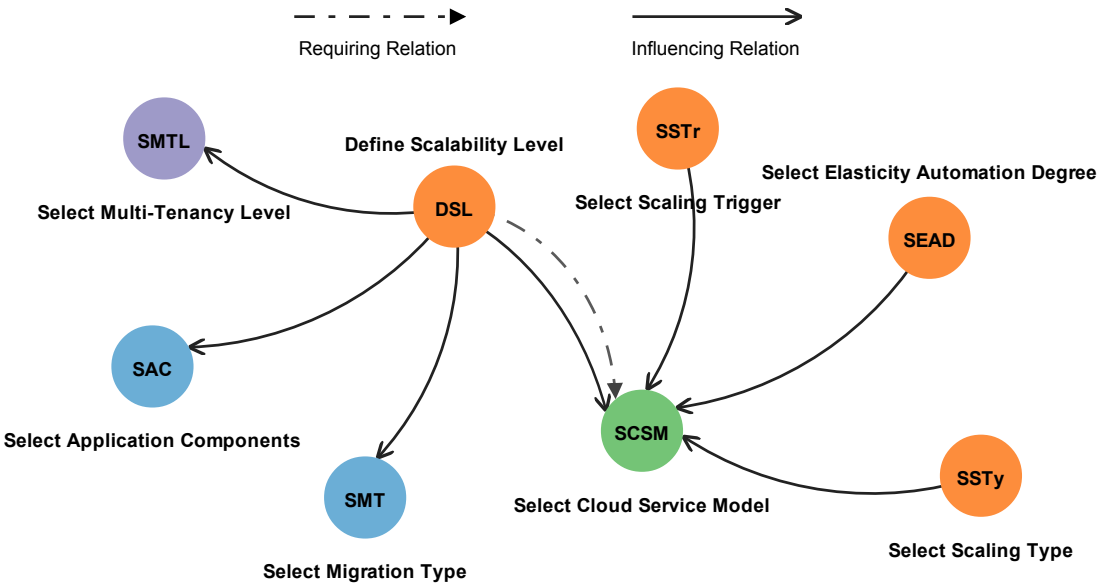

Figure 3.17.: Outgoing decision relations of *Define Elasticity Strategy*.

With respect to decision point *Select Service Provider / Offering*, six new relations have been identified. Four affecting relations between all decisions and the *Select Cloud Vendor* decision since the vendor has to support not only the level and type but also the automation and trigger options. In fact, a wide range of configurable scaling options is not necessarily common. *Microsoft Azure*[3], for example, introduced semi-automatic scaling, a common feature, roughly a year ago and still lacks behind in comparison to other cloud vendors such as *Amazon* or third-party scaling applications [84]. The other two relations are influencing associations from *Select Elasticity Automation Degree* and *Select Scaling Trigger* towards *Select Cloud Service Model*. For instance, any scaling automation and trigger would be useless in case of a SaaS environment because in that scenario the migrator would not have any direct influence on how the scaling is performed by the cloud vendor. The remaining relations can be confirmed.

A requiring relation between *Define Scalability Level* and *Select Cloud Service Model* has been introduced, see Fig. 3.17. Since the scaling level implies that the cloud consumer wants to have control over that level, an appropriate service model has to be chosen. Therefore, a selected scaling level entails a service model and cannot stand alone. Similarly to the *Define Application Distribution* decision point, the requiring relations within the *Define Elasticity Strategy* decision point will lead at least

[3] `http://azure.microsoft.com/`

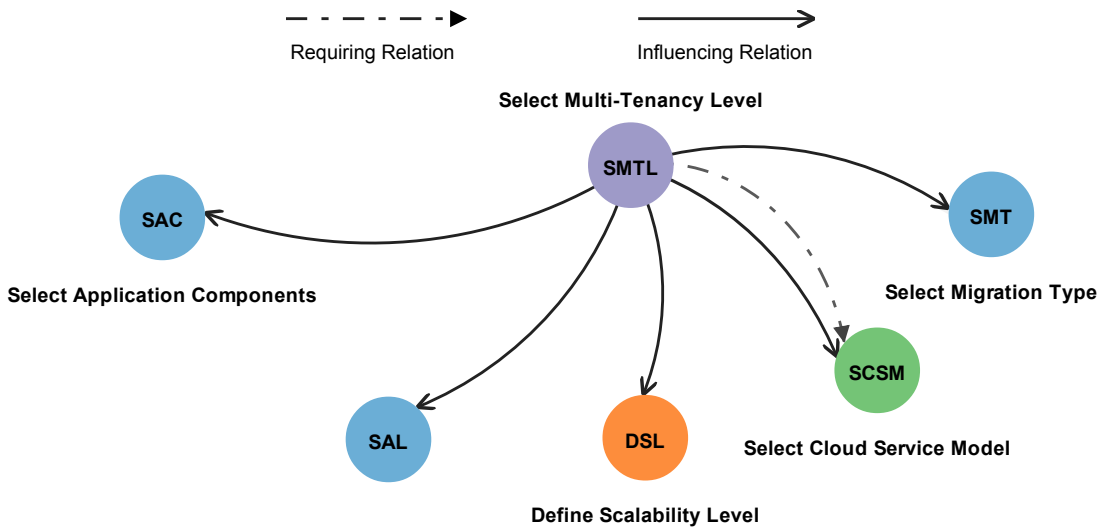

Figure 3.18.: Outgoing decision relations of *Define Multi-Tenancy Requirements*.

to a basic scaling strategy including a selected service model no matter which of its decisions is chosen first.

Relations from Define Multi-Tenancy Requirements towards other decision points: The *Select Multi-Tenancy Level* decision is the only remaining decision under this decision point. As for the previous two decision points, the influencing relations towards *Select Scaling Type* and *Select Application Tier* have been removed. All remaining influencing relations can be confirmed. Furthermore, an affecting relationship towards the *Select Cloud Vendor* decision has been introduced. According to the desired MTL, the cloud vendor has to ensure the separation between tenants. Otherwise severe problems regarding data and performance isolation could arise [10], [68], [69].

One new requiring relation from *Select Multi-Tenancy Level* towards *Select Cloud Service Model* has been identified, see Fig. 3.18. In case a MTL is selected it is essential with which service model it will be implemented since not all combinations are possible. For example, *Shared Hardware Multi-Tenancy* would not be suitable for a SaaS environment because everything might be shared among tenants solely dependent on the cloud vendor. Therefore, the desired separation cannot be enforced or ensured and will be highly unlikely [10].

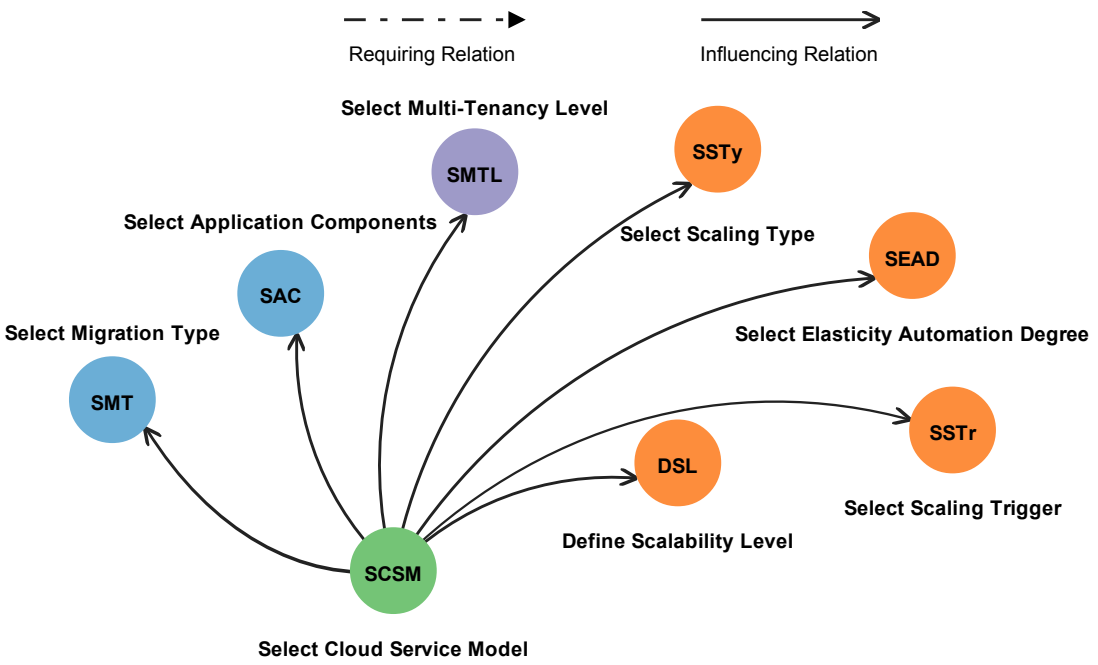

Figure 3.19.: Outgoing decision relations of *Select Service Provider / Offering*.

Relations from Select Service Provider / Offering towards other decision points: With regard to the *Define Application Distribution* decision point, as in the reverse case, an influence between *Define Cloud Hosting* and *Select Migration Type* cannot be confirmed [13]. Thus the relation has been removed. Three new influencing relations from *Select Cloud Service Model* towards *Select Scaling Type*, *Select Elasticity Automation Degree* and *Select Scaling Trigger* have been added, see Fig. 3.19. The service model influences the possible selection possibilities for any of the mentioned decisions. For instance, since the scaling decisions imply that the cloud consumer wants to have control of the desired and selected option a SaaS environment is not possible. In that case the cloud vendor would decide how the application is scaled. Hence, in order to use a certain service model it might be necessary to waive some of the scaling options which is depicted through the aforementioned relations.

Another influencing relation exists from *Select Cloud Service Model* towards *Select Application Components* since a middleware component can only be migrated with an IaaS or PaaS solution, see Section 3.1. All remaining influencing relations can be confirmed, leading to a total of seven influencing relations. All outgoing relations from the *Select Cloud Service Model* decision can be seen in Fig. 3.19.

Four new binding relations from *Select Cloud Vendor* towards all decisions of the *Define Elasticity Strategy* decision point have been added, see Fig. 3.20. A selected

vendor offers a specific set of scaling options and therefore binds the scaling related decisions. Even though it might be possible to add trigger or automation functionality on top of the provided scaling options, the possibilities are still bound by the vendor's API functionalities [53].

3.6. Summary of the Refinement

Subsequently, a summary of the undertaken changes and the results of the refinement and review will be given.

3.6.1. Identified Relations

In [18] two relationship types, namely influencing and determining, had been identified on the level of decisions. The determining relationship type no longer exists after the review, yet three new relationship types have been introduced. Affecting relations can exist from any decision towards the *Select Cloud Vendor* decision. They denote that a decision imposes certain requirements upon the cloud vendor, hence affecting the selection of the very same. In Fig. 3.20 it can be seen that *Define Resource Location* affects the *Select Cloud Vendor* decision because the cloud vendor has to comply to the specified jurisdiction. In the case a vendor cannot offer a resource location in the same jurisdiction as the company, the vendor cannot be chosen. Naturally, the complete *Define Application Distribution* decision point does not have any binding or affecting relations because the application topology and distribution is not relevant with regard to the cloud vendor.

As a pendant, binding relations can only exist from the *Select Cloud Vendor* decision towards any other decision, see Fig. 3.20. They denote that the variable participating decision is subject to the cloud vendor regarding the support of the desired feature and the actual implementation. A selected cloud vendor might offer only a specific subset of service models, hence the decision *Select Cloud Service Model* is bound by the vendor potentially reducing the amount of selectable outcomes.

An influencing relation can exist between any decision. It denotes that a selection of an outcome influences the possible selectable outcomes in the related decision. In Fig. 3.21 it is easily noted that the decisions heavily influence each other, especially the *Select Application Components*, *Select Migration Type*, *Define Scalability Level*, *Select Multi-Tenancy Level* and *Select Cloud Service Model* decisions. They can be considered essential to determine in order to migrate an application. Also, it is

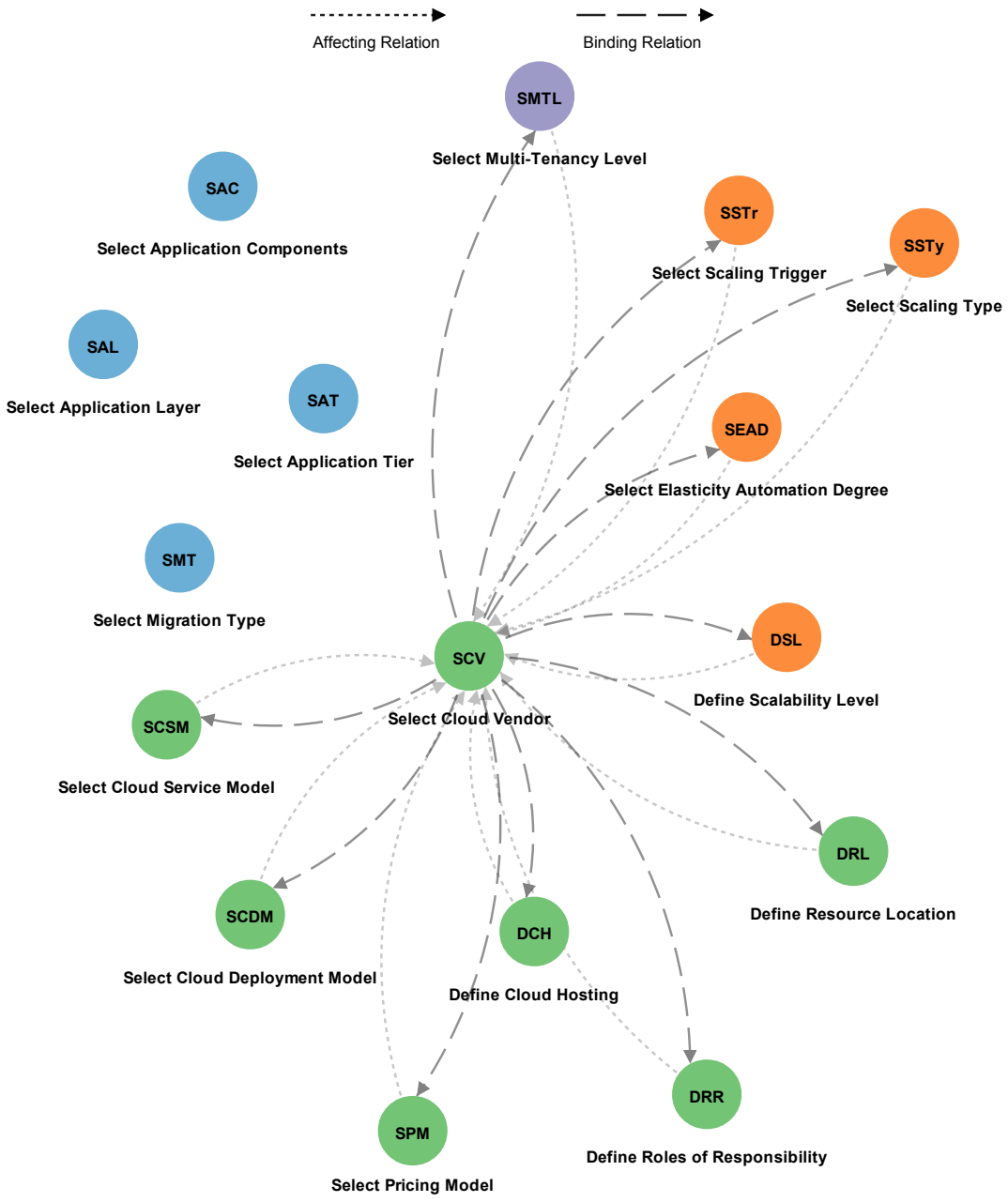

Figure 3.20.: Affecting and binding relationships between decisions.

clearly visible that only the *Select Cloud Service Model* of the *Select Service Provider / Offering* decision point has any influencing relations towards other decision points. The remaining decisions within the decision point are either without influencing relations, in the case of *Select Pricing Model* and *Select Cloud Vendor*, or influence each other thus forming a separate graph.

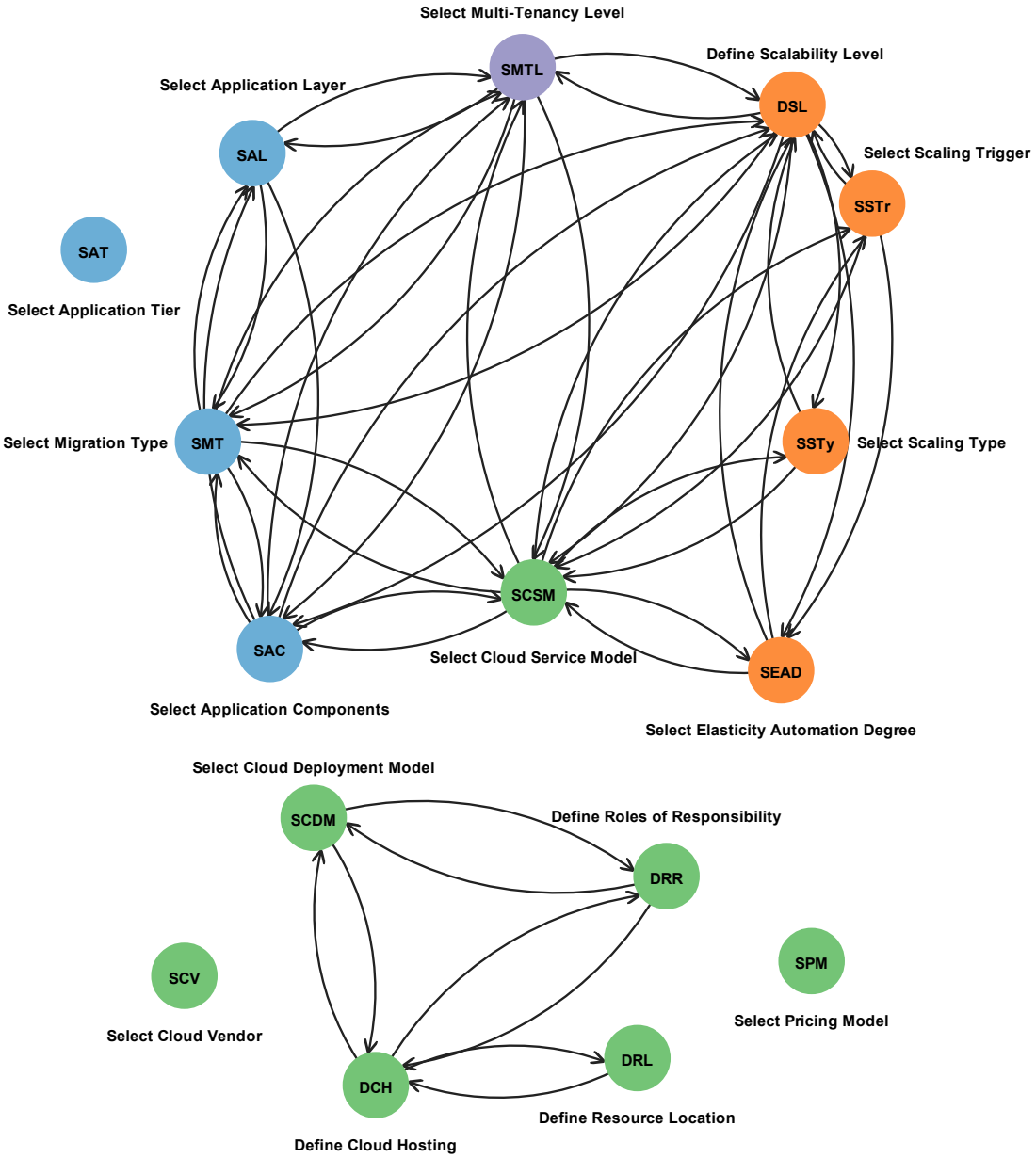

Figure 3.21.: Influencing relationships between decisions.

A requiring relation can exist between any decision. It denotes that as soon as a decision that has a requiring relation towards another decision is specified, the latter has to be specified as well. Several requiring relations are logically pointing towards the aforementioned strongly influenced decisions as can be seen in Fig. 3.22. The requiring relations within the *Define Elasticity Strategy* decision point will ensure that at least a basic and/or coherent scaling strategy is chosen. Selecting a trigger on its

own is not possible and will inevitably lead to a complete definition of all scaling decisions. Simply selecting the scaling level on the other hand is possible since the level is sufficient and one does not have to fully specify the exact scaling strategy. The same approach applies for the *Define Application Distribution* respectively.

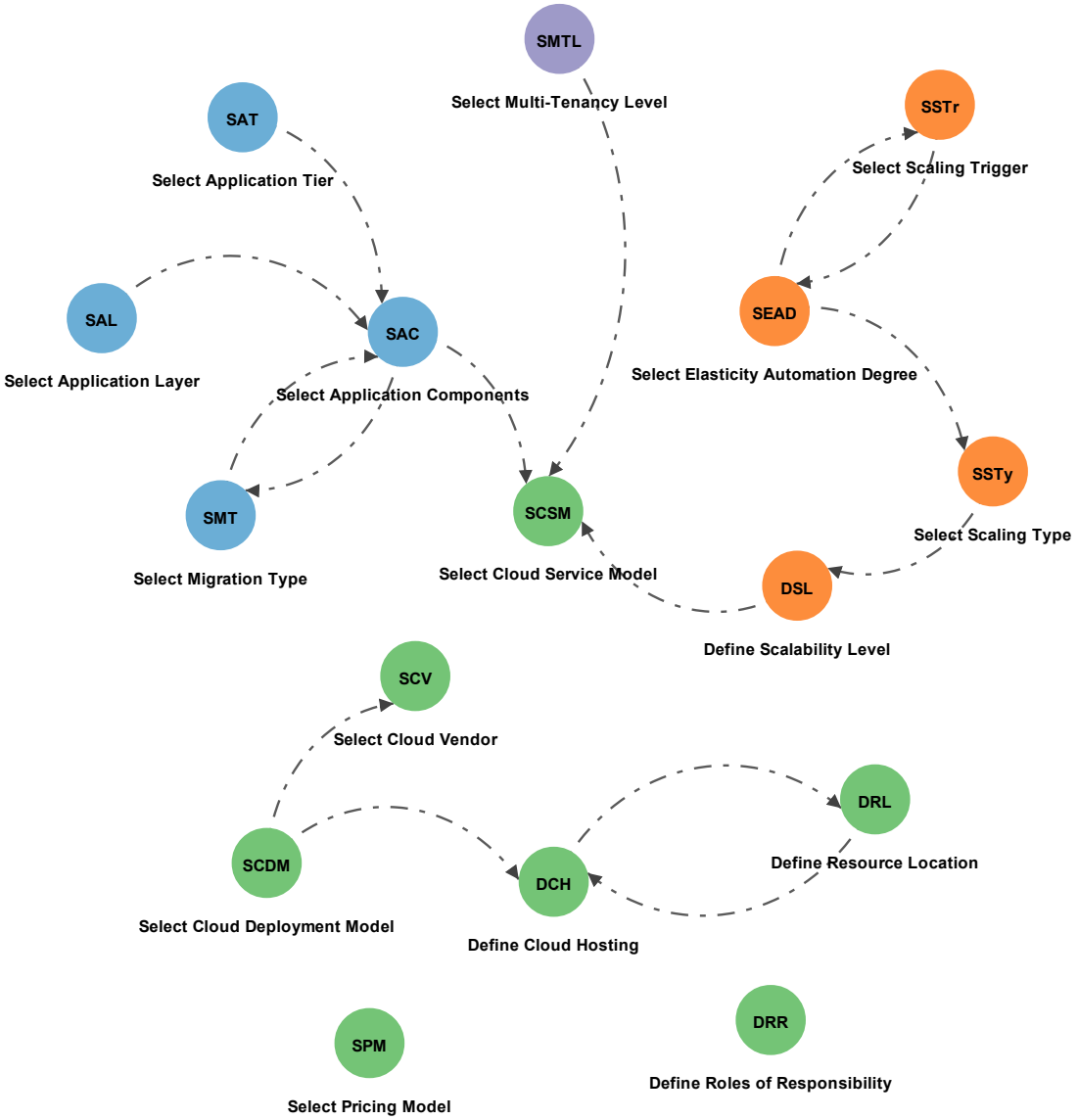

Figure 3.22.: Requiring relationships between decisions.

3.6.2. The Refined Knowledge Base

In Table 3.1 the refined knowledge base is summarized. While all decision points and, except for the *Select Kind of Multi-Tenancy* decision, all decisions still exist, the outcomes have been significantly changed.

Table 3.1.: The refined decision support framework knowledge base showing the decisions and outcomes for each decision point.

Decision Point – Define Application Distribution	
Select Application Layer	Presentation Layer Layer
	Business Layer
	Resource Layer
	Presentation + Business Layer
	Presentation + Resource Layer
	Business + Resource Layer
	Presentation + Business + Resource Layer
Select Application Tier	Client Tier
	Application Tier
	Data Tier
	Client + Application Tier
	Client + Data Tier
	Application + Data Tier
	Client + Application + Data Tier
Select Application Components	Application Component
	Application Components
	Middleware Component
	Middleware Components
	Application + Middleware Component
	Application Component + Middleware Components
	Middleware Component + Application Components
	Application + Middleware Components
Select Migration Type	Migration Type I
	Migration Type II
	Migration Type III
	Migration Type IV

Decision Point – Define Elasticity Strategy	
Define Scalability Level	No Scaling VM Level Scaling Middleware Level Scaling Application Level Scaling VM + Middleware Level Scaling VM + Application Level Scaling Middleware + Application Level Scaling VM + Middleware + Application Level Scaling
Select Scaling Type	Vertical Scaling Horizontal Scaling Hybrid Scaling
Select Elasticity Automation Degree	Manual Scaling Semi-Automatic Scaling Semi-Automatic Third-Party Scaling Automatic Scaling Automatic Third-Party Scaling
Select Scaling Trigger	No Trigger Event-Driven Trigger Proactive Trigger
Decision Point – Define Multi-Tenancy Requirements	
Select Multi-Tenancy Level	Shared Hardware Multi-Tenancy Shared OS Multi-Tenancy Shared Middleware Multi-Tenancy Shared Application Multi-Tenancy
Decision Point – Select Service Provider / Offering	
Select Cloud Deployment Model	Public Cloud Private Cloud Community Cloud Hybrid Cloud
Select Cloud Service Model	IaaS PaaS SaaS IaaS + PaaS IaaS + SaaS

	PaaS + SaaS
	Iaas + PaaS + SaaS
Define Cloud Hosting	On-Premise Hosting
	Off-Premise Hosting
	Hybrid Hosting
Define Roles of Responsibility	Inhouse
	Management
	Outsourced
	Inhouse + Management
	Inhouse + Outsourced
	Management + Outsourced
	Inhouse + Management + Outsourced
Select Cloud Vendor	Evaluated Cloud Vendor
Select Pricing Model	Free
	Pay-Per-Use
	Pay-Per-Unit
	Charge-Per-Use (Subscription)
Define Resource Location	Data In Same Jurisdiction
	Data In Different Jurisdiction

To depict the changes to the knowledge base, the number of decision points, decisions and outcomes per decision point have been quantified in Table 3.2. In this regard, four decisions, namely *Select Application Components*, *Define Scalability Level*, *Select Multi-Tenancy Level* and *Define Resource Location*, have been deemed as modified since their outcomes and general understanding have been significantly altered in contrast to their predecessors. This also means that their old outcomes are considered as removed, and their new outcomes as added instead of modified. Modified on the level of outcomes is defined as adjustments such as slight changes to the underlying assumptions.

Only one decision, *Select Kind of Multi-Tenancy*, and none of the decision points have been deleted. Previously, 67 different basic outcomes, i.e., outcomes that cannot be expressed by a combination of others, have been reduced to 49. However, due to the removal of 29 basic outcomes under the *Select Multi-Tenancy Architecture* decision without actually losing much information, the numbers of basic outcomes for other decisions have been increased. Even though the total number of outcomes have therefore been only merely reduced by seven, in fact 45 outcomes have been deleted and 38 new ones added leading to a total of 39 out of 84 outcomes remaining the

same as previously defined in [18]. Hence, roughly 54% of all original outcomes have been discarded or substituted by different outcomes which in turn means that only 46% of the previous knowledge base remains the same.

Table 3.2.: Summary of the changes between the previous (CloudDSF) and the refined knowledge base (CloudDSF+).

	CloudDSF	**Removed**	**Added**	**Modified**	**CloudDSF+**
No. of Decision Points	**4**	**0**	**0**	**0**	**4**
No. of Decisions	**17**	**1**	**0**	**4**	**16**
No. of Outcomes per Decision Point					
Define Application Distribution	14	4	16	5	26
Define Elasticity Strategy	14	3	8	2	19
Define Multi-Tenancy Requirements	31	31	4	0	4
Select Service Provider / Offering	25	7	10	6	28
Total No. of Outcomes	**84**	**45**	**38**	**13**	**77**

The number of relations between decisions (within decision points and between them) has been summarized in Table 3.3. Within decision points, the number of relations has decreased except for the decision point *Select Service Provider / Offering* where, due to the *Select Cloud Vendor* decision, several affecting and binding relations have been added. Most relations between decision point have been removed starting from decisions belonging to the *Define Application Distribution* decision point and among those, especially from the *Select Application Tier* decision. In total, the number of relations between decision points has increased due to the new affecting, binding and requiring relations. In fact 30 influencing decisions have been removed and only 14 new decisions have been added.

Through the undertaken refinements a more detailed selection per decision is now feasible since all different possible combinations of basic outcomes are denoted by one corresponding outcome instead of one generic outcome as before. Also, the expressiveness and conciseness of the knowledge base has been significantly increased while simultaneously ensuring the suitability of the outcomes for an identification of relations between them. The review of relations between decisions not only ensures the consistency and validity of the knowledge base but will facilitate and also serve as a basis for the following elaboration of relations between outcomes in the subsequent chapter.

Table 3.3.: Quantification of decision relations before and after the review.

	CloudDSF	Removed	Added	Confirmed	CloudDSF+
No. of Relations Within Decision Points					
Define Application Distribution	12	7	5	5	10
Define Elasticity Strategy	4	0	8	4	12
Define Multi-Tenancy Requirements	2	2	0	0	0
Select Service Provider / Offering	11	5	18	6	24
No. of Relations Between Decision Points					
Define Application Distribution	17	10	1	7	8
Define Elasticity Strategy	10	6	8	4	12
Define Multi-Tenancy Requirements	7	2	2	5	7
Select Service Provider / Offering	4	1	9	3	12
Total	**67**	**33**	**51**	**34**	**85**
No. of Occurrences of Relationship Types					
Affecting	0	0	11	–	11
Binding	0	0	11	–	11
Determining	3	3	0	–	0
Influencing	64	30	14	–	48
Requiring	0	0	15	–	15

4. Extension of the Decision Support Framework With Relations Between Outcomes

In the following chapter the relations among outcomes between different decisions will be elaborated and defined based on the previously refined and updated knowledge base. This results in considerably extending the decision support framework. Naturally, only those combinations of outcomes that already have an identified relation between their respective decisions will be considered, see Section 3.5. In order to keep the chapter less verbose, all affecting and binding relationships will be solely discussed under the *Select Cloud Vendor* decision since a further refinement on the level of outcomes is impossible, see Section 4.4.5. Within the context of the elaboration, several different relationship types between outcomes have been discovered and are defined beforehand in Table 4.1 to facilitate a clear and concise discussion. The table also depicts the used abbreviations and color scheme for all subsequent tables that denote the relations between outcomes including those in Appendix A.1.

It is critical to mention that several assumptions have been made. Firstly, within each decision only one outcome can be selected at any time equaling an XOR (exclusive or) relationship between them. Secondly, several decisions consist of basic outcomes and combinatorial outcomes that can be expressed as a combination of the former. *The combinatorial outcomes inherit the relations of their basic outcomes.* However, in case the contributing basic outcomes have a contradicting relation towards another outcome the prevailing relation will be determined on a per decision basis.

The following sections each correspond to one specific decision point and are structured by their respective decisions. For each decision all relations from its outcomes towards other relevant outcomes are discussed and defined.

Table 4.1.: Definition of the relationship types between outcomes.

Relationship Type	Abbrev.	Definition
Allowing	a	An allowing relation between outcome A and outcome B denotes that in case A is selected, B can be selected as well. Consequently, A neither entails nor prohibits B.
Excluding	ex	An excluding relation between outcome A and outcome B denotes that if A is selected, B can no longer be selected anymore. Hence, A prohibits B.
Including	in	An including relation between outcome A and outcome B denotes that if A is selected, B becomes obligatory and has to be selected as well. Hence, A entails B.
Affecting	aff	An affecting relation can only exist from outcomes of any decision towards the *Evaluated Cloud Vendor* outcome of the *Select Cloud Vendor* decision. It denotes that the participating outcome imposes certain requirements upon the cloud vendor, hence affecting the selection of the vendor per se.
(Externally) Binding	eb	An (externally) binding relation can only exist from the outcome *Evaluated Cloud Vendor* of the *Select Cloud Vendor* decision towards any other decision's outcomes. It denotes that the variable participating outcome is subject to the cloud vendor regarding the support of the desired feature and the actual implementation. Thus, it is externally bounded and out of the influence of the migrator.

4.1. Define Application Distribution

All of in the following identified relations are stated in detail in the appendix from Table A.3 to Table A.7.

4.1.1. Select Application Layer

Select Application Components: The basic outcome *Presentation Layer* can comprise one *Application Component* or multiple *Application Components*. It can also contain one or more *Middleware Components* but only if at least one application component is present. Even though web applications might merge presentation and business layer logic, thus increasing the middleware capabilities and responsibilities of the presentation layer, the absence of an application component is deemed highly unlikely [51]. Therefore, and to further differentiate the *Presentation Layer* from the

Table 4.2.: Subset of outcome relations of *Select Application Layer*.

Select Application Layer	Select Application Components								Select Migration Type			
	Application Component	*Application Components*	*Middleware Component*	*Middleware Components*	*Application + Middleware Component*	*Application Component + Middleware Components*	*Middleware Component + Application Components*	*Application + Middleware Components*	*Migration Type I*	*Migration Type II*	*Migration Type III*	*Migration Type IV*
Presentation Layer	a	a	ex	ex	a	a	a	a	a	a	ex	ex
Business Layer	a	a	a	a	a	a	a	a	a	a	ex	ex
Resource Layer	ex	ex	a	a	ex	ex	ex	ex	a	a	ex	ex
Presentation + Business Layer	ex	a	ex	ex	a	a	a	a	ex	a	ex	ex
Presentation + Resource Layer	ex	ex	ex	ex	a	a	a	a	ex	a	ex	ex
Business + Resource Layer	ex	ex	ex	a	a	a	a	a	ex	a	ex	ex
Presentation + Business + Resource Layer	ex	ex	ex	ex	ex	a	a	a	ex	a	a	a

Business Layer outcome, two excluding relationships for the respective middleware component outcomes have been defined, see Table 4.2.

The outcome *Business Layer* can contain any combination of components, whereas the *Resource Layer* can only contain middleware components leading to six excluding relationships and two allowing relationships towards *Middleware Component* and *Middleware Components*. Any combination of two basic outcomes logically renders a single component impossible but allows any of the other combinatorial outcomes. Also, the dual combinations including the basic outcome *Resource Layer* prohibit a selection of the *Application Components* outcome. For the outcome *Presentation + Business + Resource Layer* the same applies with one exception. The outcome excludes *Application + Middleware Component* since three layers entail at least three components.

Select Migration Type: The three basic outcomes allow (due to the fact that they can contain single or multiple components) either *Migration Type I* or *Migration Type II* and exclude the other two. Any combination of two basic outcomes allows only *Migration Type II* and excludes all others since logically more than one layer and thus

more than one component is migrated, yet not the whole application, see Table 4.2. Hence, the combinatorial outcome *Presentation + Business + Resource Layer* allows, besides *Migration Type II*, also *Migration Type III* as well as *Migration Type IV* and fulfills their requirement that the complete application is migrated. This is naturally the case since in accordance with the definition in Section 3.1 all components on a selected layer have to be migrated. In this regard it can also be taken for granted that a complete application migration encompasses application as well as middleware components.

Select Multi-Tenancy Level: Conforming to Section 3.3, the *Select Multi-Tenancy Level* decision and therefore all its outcomes requires the complete application to be migrated in order to be applicable. This leads to excluding relationships towards all MTLs except for the outcome *Presentation + Business + Resource Layer* that denotes a complete application migration, thus leading to an allowing relationship, see Table A.3.

4.1.2. Select Application Tier

No relations exists between any decision and the *Select Application Tier* decision following the discussion in Section 3.1. Consequently, no relations have to be identified on the level of outcomes.

4.1.3. Select Application Components

Select Application Layer: The relations from the outcomes of the *Select Application Components* decision towards the *Select Application Layer* decision's outcomes are the inversion of the relation in the opposite direction, see Table A.4, as described in Section 4.1.1. The same arguments and reasoning applies and will therefore not be repeated.

Select Migration Type: Regarding the outcome of the *Select Migration Type* decision, see Table 4.3, *Migration Type I* is only possible with the outcome *Application Component* or *Middleware Component* (allowing relations) whereas all others exclude it. *Migration Type II* is excluded by the two single component type outcomes but is allowed by any other outcome. *Migration Type III* and *Migration Type IV* are excluded

Table 4.3.: Subset of outcome relations of *Select Application Components*.

Select Application Components	Select Migration Type				Select Cloud Service Model						
	Migration Type I	*Migration Type II*	*Migration Type III*	*Migration Type IV*	*IaaS*	*PaaS*	*SaaS*	*IaaS + PaaS*	*IaaS + SaaS*	*PaaS + SaaS*	*Iaas + PaaS + SaaS*
Application Component	a	ex	ex	ex	a	a	a	ex	ex	ex	ex
Application Components	ex	a	ex	ex	a	a	a	a	a	a	a
Middleware Component	a	ex	ex	ex	a	a	ex	ex	ex	ex	ex
Middleware Components	ex	a	ex	ex	a	a	ex	a	ex	ex	ex
Application + Middleware Component	ex	a	ex	ex	a	a	ex	a	ex	ex	ex
Application Component + Middleware Components	ex	a	ex	ex	a	a	ex	a	ex	ex	ex
Middleware Component + Application Components	ex	a	ex	ex	a	a	ex	a	ex	ex	ex
Application + Middleware Components	ex	a	a	a	a	a	ex	a	ex	ex	ex

by all outcomes except the *Application + Middleware Components* outcome. Theoretically, the outcomes *Application Component + Middleware Components* and *Middleware Component + Application Components* would be possible as well. However, in the case that the complete application is migrated it does not influence further decision making if the application comprises only a single component of any of the two component types. The outcome *Application + Middleware Components* denotes therefore a migration of the complete application whereas the former two outcomes are denoting more detailed cases of *Migration Type II* migrations.

Define Scalability Level: The outcomes *Application Component* and *Application Components* on their own logically exclude any outcome including *Middleware Level Scaling* because no middleware is migrated in the first place. This applies for the outcomes *Middleware Component* and *Middleware Components* towards *Application Level Scaling* respectively. *VM Level Scaling* is only possible if some sort of middleware is migrated. Nevertheless, all combinatorial outcomes allow any form of scaling since a component is always present to which a specific scaling level can be applied. For instance, the outcome *Application Component + Middleware Components* allows, through the comprising application component, *Application Level Scaling*. However, a scaling of the complete application stack denoted by *VM + Middleware + Application Level Scaling* would also be possible, see Table A.5. With regard to the *No Scaling* outcome no restrictions apply, thus all outcomes have an allowing relationship towards it. The migrator always has the freedom of decision if the migrated

application components shall be scaled or not, assuming that an appropriate mechanism is available, see Section 4.2.1.

Select Multi-Tenancy Level: Similar to the relations between outcomes of the *Select Application Layer* and *Select Multi-Tenancy Level* decision only the outcome *Application + Middleware Components*, depicting a complete application migration, allows multi-tenancy and thus any of the MTLs. All other outcomes of the *Select Application Components* decision exclude any form of multi-tenancy, see Table A.5.

Select Cloud Service Model: The relations between components and service models are directly based on the assumptions stated in Section 3.1 and Section 3.5.1. Application components can use any service model whereas middleware components are restricted to IaaS or PaaS. Logically, a single component can only be migrated using one service model leading to excluding relationships from *Application Component* and *Middleware Component* towards all combinatorial outcomes and for the latter also towards *SaaS*, see Table 4.3. Multiple application as well as middleware components and any combination of them can be migrated using *IaaS*, *PaaS* or *IaaS + PaaS*.

The *SaaS* outcome is only applicable for application components, hence only for two outcomes of the *Select Application Components* decision and for none of the combinatorial ones. In those cases, any allowing relation towards *SaaS* is prevailed by the excluding relation from the outcomes *Middleware Component* or *Middleware Components* respectively. In actuality, using SaaS equals a substitution of the migrated functionality or application. A suitable SaaS solution is highly unlikely for any major legacy application. Besides, typical middleware components such as databases or application servers are also only offered as part of PaaS solutions since on their own they do not per se deliver any form of business functionality, hence rendering themselves unable to be subsumed under SaaS [10]. Moreover, migrating an application component to a SaaS environment would render the underlying middleware unnecessary in the first place as well as requiring the migration of middleware simultaneously.

4.1.4. Select Migration Type

Select Application Layer: All relations from the outcomes of *Select Migration Type* towards the outcomes of *Select Application Layer* are a reflection as for the previously defined relations in reverse, see Section 4.1.1. However, two relations namely those

from *Migration Type III* and *Migration Type IV* towards the outcome *Presentation + Business + Resource Layer* have been defined as including instead of allowing relations, see Table A.6. Both types depict a migration of the complete application thus all layers must be included.

Select Application Components: The same reasoning as above also applies with regard to the relations towards the *Select Application Components* decisions. Therefore, all relations between outcomes are the same as in reverse, see Section 4.1.3 and Table A.6 respectively. The only exceptions are the two relations from *Migration Type III* and *Migration Type IV* towards the outcome *Application + Middleware Components*. Both relations have been substituted by an including relationship.

Define Scalability Level: *Migration Type I* has, since only one component is migrated, the relations resulting out of the combination of defined allowing relations from *Application Component* and *Middleware Component* towards the *Define Scalability Level* decision as defined in Section 4.1.3. Therefore, five allowing and three excluding relations have been identified. *Migration Type II* and *Migration Type III* allow any form of scaling, whereas *Migration Type IV* excludes the *No Scaling* option. As defined in Section 3.1, the outcome *Migration Type IV* explicitly attempts to leverage cloud computing capabilities, thus including some level of scaling, see Table A.7.

Select Multi-Tenancy Level: Multi-tenancy is, as already mentioned, only possible if the complete application is migrated. Therefore *Migration Type I* and *Migration Type II* exclude any outcome of the *Select Multi-Tenancy Level* decision, see Table 4.4. For instance, in the case of migrating a database to a public cloud, it is highly likely that multi-tenancy is applied through the cloud vendor for the underlying resources. However, it would only apply to that specific component of the partially migrated application and therein out of the scope and control of the consumer. Therefore, it contradicts the definition and understanding of multi-tenancy as stated in Section 3.3. This circumstance can only be partly reflected by the decision support framework through the binding relations from the cloud vendor.

Migration Type III allows any MTL due to the fact that by definition the complete application is migrated, wrapped as-is in a VM. Therefore, extensive reengineering might be necessary to implement multi-tenancy on a higher level depending on the application architecture [13]. The same relations apply for *Migration Type IV* that specifically tries to leverage multi-tenancy. In the end, *Migration Type IV* will at least

Table 4.4.: Subset of outcome relations of *Select Migration Type*.

Select Migration Type	Select Multi-Tenancy Level				Select Cloud Service Model						
	Shared Hardware Multi-Tenancy	*Shared OS Multi-Tenancy*	*Shared Middleware Multi-Tenancy*	*Shared Application Multi-Tenancy*	*IaaS*	*PaaS*	*SaaS*	*IaaS + PaaS*	*IaaS + SaaS*	*PaaS + SaaS*	*IaaS + PaaS + SaaS*
Migration Type I	ex	ex	ex	ex	a	a	a	ex	ex	ex	ex
Migration Type II	ex	ex	ex	ex	a	a	a	a	a	a	a
Migration Type III	a	a	a	a	in	ex	ex	ex	ex	ex	ex
Migration Type IV	a	a	a	a	ex	a	a	ex	ex	a	ex

use *Shared Middleware Multi-Tenancy* or higher due to the restrictions regarding the service model, see Section 4.4.2. However, this restriction is not in the scope of the discussion of the relations between these two decisions.

Select Cloud Service Model: In Section 3.1, the correlations between the migration types and service models have been stated. More specifically, *Migration Type I* and *Migration Type II* can be used with any of the service models leading to allowing relations. The latter can also be used with any combination of service models since multiple components are migrated, see Table 4.4. By definition, *Migration Type III* needs the *IaaS* outcome whereas *Migration Type IV* can either use *PaaS*, *SaaS* or *PaaS + SaaS* [13].

4.2. Define Elasticity Strategy

All identified relations in the following are stated in detail in the appendix from Table A.8 to Table A.12.

4.2.1. Define Scalability Level

In advance it can be noted that the *No Scaling* outcome naturally renders any further specification through other decisions under the *Define Elasticity Strategy* decision

point useless. If nothing is selected to be scaled then any form of scaling type, automation or trigger is no longer applicable anymore. Therefore, from the *No Scaling* outcome excluding relations have been defined towards all outcomes of the aforementioned decisions, see Tables A.8 and A.9, and will not be explicitly stated in the following.

Select Application Components and Select Migration Type: The relations between outcomes from *Define Scalability Level* towards the *Select Application Components* decision, see Section 4.1.3, are an exact reflection and will not be stated in the following. The same applies for the relation towards the *Select Migration Type* decision, see Table A.8.

Select Scaling Type: In accordance with Section 3.2, vertical scaling on its own is only applicable for VMs. Logically, only the outcome *VM Level Scaling* allows *Vertical Scaling*. *Horizontal Scaling* is allowed by any outcome of the *Define Scalability Level* decision. *Hybrid Scaling*, the combination of both scaling types, is only applicable for scaling levels subsuming the *VM Level Scaling* thereby satisfying the vertical scaling portion, see Table 4.5. The same argument but probably more intuitive applies for the other way around, see Section 4.2.2.

Select Elasticity Automation Degree: The relations are exactly the same as in reverse. Through the more comprehensible description from the *Select Elasticity Automation Degree* decision point of view, the relations have been described under Section 4.2.3.

Select Scaling Trigger: As previously mentioned, the *No Scaling* outcome excludes any of the scaling triggers, see Table A.9. Besides that, only allowing relationships exist. In fact, the scaling level can be described as agnostic towards the trigger. Any further restrictions that might apply will be denoted by other decisions such as *Select Elasticity Automation Degree* or *Select Cloud Service Model* and are not in the scope of this decision.

Select Multi-Tenancy Level: The relations between the outcome of the *Define Scalability Level* decision towards outcomes of the *Select Multi-Tenancy Level* decision are an one-to-one reflection as for the other way around, see Table A.9 and Table A.11 respectively following the discussion in Section 4.3.1.

Table 4.5.: Subset of outcome relations of *Define Scalability Level*.

	Vertical Scaling	Horizontal Scaling	Hybrid Scaling	IaaS	PaaS	SaaS	IaaS + PaaS	IaaS + SaaS	PaaS + SaaS	IaaS + PaaS + SaaS
Define Scalability Level	**Select Scaling Type**			**Select Cloud Service Model**						
No Scaling	ex	ex	ex	a	a	ex	a	a	a	a
VM Level Scaling	a	a	a	a	ex	ex	a	a	ex	a
Middleware Level Scaling	ex	a	ex	a	ex	ex	a	a	ex	a
Application Level Scaling	ex	a	ex	a	a	ex	a	a	a	a
VM + Middleware Level Scaling	ex	a	a	a	ex	ex	a	a	ex	a
VM + Application Level Scaling	ex	a	a	a	ex	ex	a	a	ex	a
Middleware + Application Level Scaling	ex	a	ex	a	ex	ex	a	a	ex	a
VM + Middleware + Application Level Scaling	ex	a	a	a	ex	ex	a	a	ex	a

Select Cloud Service Model: The service models are predominantly distinguished by their separation of control between provider and consumer [19], with IaaS providing the most control and SaaS the least. With regard to scaling, this translates as follows. In case of the *IaaS* outcome, the consumer has full control of all parts including the VMs supplied by the provider. Thus, any scaling level outcome or none at all can be implemented by the consumer leading to allowing relations only. SaaS on the other hand "frees" the consumer from any sort of scaling in the first place. Logically any outcome of the *Define Scalability Level* decision excludes *SaaS* as selectable outcome since it is completely out of the scope and control of the consumer. Therefore, in order to select a pure SaaS solution no scaling level must be chosen.

PaaS as the service model in between the former two removes both the *VM Level Scaling* and *Middleware Level Scaling* outcomes out of the scope of the consumer but still allows *Application Level Scaling* on top of the provided PaaS environment. PaaS offerings that allow influence regarding VMs of the provider's underlying IaaS solution cannot be considered fully-fledged to enable *VM Level Scaling* and are also very rare in practice. In fact, *Jelastic* claims to be the only provider supporting true vertical scalability for PaaS [85]. However, this special case rather mimics a combination of a IaaS and PaaS offering and furthermore several problems remain [85]. To that end, only the *Application Level Scaling* and the *No Scaling* outcomes have an allowing relationship towards the outcome *PaaS*, see Table 4.5. The *No Scaling* option is valid since one does not have to scale the application as a whole even though the underlying resources might be scaled by the cloud provider. The outcome still fulfills its purpose and renders a further refinement of the scaling strategy unnecessary.

Towards all combinatorial outcomes including the *IaaS* option the same relations as for the basic outcome itself applies, thus only allowing relations. Similarly, in case of *PaaS + SaaS* the same relationships from the scaling levels as towards the *PaaS* outcome apply, see Table 4.5.

4.2.2. Select Scaling Type

Define Scalability Level: It needs to be mentioned that a selected scaling type implies that some sort of scaling is desired, thus leading to excluding relations from any outcome towards the *No Scaling* option. In accordance with Section 3.2 and Section 4.2.1, all relations are a reflection as for the reverse case. Therefore, *Vertical Scaling* alone is only applicable for VMs hence the outcome only includes the *VM Level Scaling* option while excluding all others. *Horizontal Scaling* on the other hand is applicable to any scaling level thus allowing all of them. The combination of both, *Hybrid Scaling* can be applied to all scaling levels containing the *VM Level Scaling* outcome because in that case the vertical scaling part can be satisfied. Besides, restricting the *Hybrid Scaling* outcome to the *VM Level Scaling* outcome would be counterintuitive for decision makers since a differentiation of the applied scaling type per level is often not considered and rather generalized, thus vertical scaling of the underlying VMs is often implied.

Select Cloud Service Model: The different scaling types influence the possible service models similar as the *Define Scalability Level* decision does. True control over *Vertical Scaling* is only offered by a IaaS environment, see Section 4.2.1. Therefore, *Vertical Scaling* excludes *PaaS*, *SaaS* and their combination. All other outcomes are allowed. *Horizontal Scaling* additionally allows *PaaS* and *PaaS + SaaS* leading to only one excluding relationship towards *SaaS*. *Hybrid Scaling* has the exact relation as *Vertical Scaling* since the *IaaS* outcome is the determining factor to satisfy the *Vertical Scaling* outcome, see Table A.13.

4.2.3. Select Elasticity Automation Degree

Define Scalability Level: Similar to the case of the *Select Scaling Type* decision, see Section 4.2.2, any form of chosen automation indicates some sort of desired scaling thus excluding the *No Scaling* option. Furthermore, the scaling automation applies to all scaling levels uniformly. *Manual Scaling*, *Semi-Automatic Scaling* and *Semi-Automatic Third-Party Scaling* allow all different scaling levels. Even though

the semi-automatic scaling option is often sufficient and easier to apply, more demanding applications are in need of more sophisticated automated scaling strategies [86].

In CloudDSF, automatic scaling, or that which does not require any form of manually set thresholds with a proactive trigger, was considered to be in ongoing research [18]. This can partly be confirmed since most of the automatic scaling approaches in combination with a proactive (predictive) trigger, used to anticipate future load spikes, only deal with the infrastructure level and are still in their infancy [87], [88]. In addition, the term automatic scaling was and in fact is still widely used for semi-automatic scaling by providers and consumers alike and common classification schemes as well as benchmarks for scaling approaches are still missing [88].

However, extensive research is currently being performed with regard to auto scaling mechanisms by third parties based on cloud vendor offerings or open source implementations [88], [89], [90], [91], [92] also including some for higher service models [93], [94], [95]. To that end, the *Automatic Scaling* outcome, thus indicating that the scaling is performed by the migrator, is restricted due to the associated difficulties to the *VM Level Scaling* and excludes all other outcomes, see Table 4.6. The *Automatic Third-Party Scaling* outcome on the other hand allows all scaling levels, leaving the concrete implementation to the third party. How the third party actually achieves and promotes the provided scaling is out of the decision maker's concern. In fact, from the migrator point of view the relevant aspect is that an automatic scaling without direct involvement of himself is performed and ensured. This leads to seven allowing and one excluding relationship for the *Automatic Third-Party Scaling* outcome, see Table 4.6.

Select Scaling Trigger: As stated in Section 3.2, *Manual Scaling* does not describe a day-to-day activity but rather a longhand planned scaling action. Therefore, any trigger that would initiate some scaling is excluded leading to one including relation to the *No Trigger* outcome and excluded all other outcomes. The *Semi-Automatic Scaling* outcome is always paired with an *Event-Driven Trigger*. Usually defined thresholds such as memory usage, degree of CPU utilization or number of incoming requests will trigger some sort of previously defined scaling action [58]. This combination is widespread and commonly supported by the majority of cloud vendors [58], [87]. *Semi-Automatic Scaling* therefore includes the *Event-Driven Trigger* and excludes all other outcomes.

As a result of the aforementioned research activities regarding automatic scaling strategies, hybrid approaches emerged that use reactive as well as predictive triggers

Table 4.6.: Subset of outcome relations of *Select Elasticity Automation Degree*.

Select Elasticity Automation Degree	Define Scalability Level								Select Scaling Trigger		
	No Scaling	*VM Level Scaling*	*Middleware Level Scaling*	*Application Level Scaling*	*VM + Middleware Level Scaling*	*VM + Application Level Scaling*	*Middleware + Application Level Scaling*	*VM + Middleware + Application Level Scaling*	*No Trigger*	*Event-Driven Trigger*	*Proactive Trigger*
Manual Scaling	ex	a	a	a	a	a	a	a	in	ex	ex
Semi-Automatic Scaling	ex	a	a	a	a	a	a	a	ex	in	ex
Semi-Automatic Third-Party Scaling	ex	a	a	a	a	a	a	a	ex	ex	ex
Automatic Scaling	ex	a	ex	ex	ex	ex	ex	ex	ex	a	a
Automatic Third-Party Scaling	ex	a	a	a	a	a	a	a	ex	ex	ex

[88], [90]. Therefore, *Automatic Scaling* does allow the *Proactive Trigger* as well as the *Event-Driven Trigger* outcome relaxing the one-to-one relationship between automatic scaling and proactive trigger defined in [18]. Furthermore, the *Automatic Third-Party Scaling* but also the *Semi-Automatic Third-Party Scaling* render any trigger selection unnecessary since the scaling is performed by a third party. How the third party is triggering their specific scaling actions is, intentionally by selecting the respective outcome, out of interest for the consumer. Therefore, excluding relationships have been defined from those two outcomes, see Table 4.6.

Select Cloud Service Model: The relations between *Select Elasticity Automation Degree* and *Select Cloud Service Model* are straightforward based on the previously stated assumptions and Section 3.2. The outcomes *Manual Scaling*, *Semi-Automatic Scaling* and *Semi-Automatic Third-Party Scaling* all have the same relations allowing all outcomes excluding only the *SaaS* option, see Table A.11. *Automatic Scaling* requires, due to the focus on the VM level the *IaaS* outcome. Therefore, only those outcomes are allowed that include the *IaaS* outcome excluding the remaining three outcomes. The third party option does similarly, as for the other decisions before, relax that assumption towards *PaaS* leading to only one remaining excluding relationship towards *SaaS*.

The *SaaS* outcome is always excluded since in that case any scaling is out of the

control of the consumer. For all the combinatorial outcomes containing the *SaaS* or maybe also *PaaS* option will naturally not be entailed by the automation but bound by the cloud vendor. However, this partially bound relation cannot be explicitly depicted and is a limitation of the decisions support framework.

4.2.4. Select Scaling Trigger

Define Scalability Level: All relations are exact reflections of the identified relations towards this outcome, see Section 4.2.1 and Table A.12 respectively. This leads to an excluding relation for each trigger towards *No Scaling* and allowing relations towards all other outcomes. The trigger is not concerned what level is scaled but that at least one level is scaled.

Select Elasticity Automation Degree: Apart from one relation from *Event-Driven Trigger* towards *Semi-Automatic Scaling* all relations are exactly the same as in the reverse case, see Section 4.2.3. The mentioned relation has been defined as allowing instead of including since an event-driven trigger can be used in conjunction with automatic as well as semi-automatic scaling.

Select Cloud Service Model: Any trigger, if chosen, implies that the cloud migrator wants to utilize that specific trigger. Thus, in the case of the *Proactive Trigger* it is only applicable for the basic outcome *IaaS* or any of the outcomes including it. All other outcomes are logically excluded. The *No Trigger* and *Event-Driven Trigger* outcomes allow any form of service model outcome except the basic outcome *SaaS*. The outcome *No Trigger* option excludes *SaaS* since the choice still indicates that some scaling is applied just not in a triggered fashion. See Section 3.4 and Table A.12 for reference.

4.3. Define Multi-Tenancy Requirements

All identified relations in the following are stated in detail in the appendix in Table A.14 and Table A.15.

4.3.1. Select Multi-Tenancy Level

Select Application Layer, Select Application Components and Select Migration Type: All outcomes of the *Select Multi-Tenancy Level* decision have an including relation towards the *Presentation + Business + Resource Layer* outcome because of the prerequisite that the complete application has to be migrated, thus all layers have to be chosen. As a consequence, excluding relationships exist towards all other outcomes. Similarly, this applies to the *Select Application Components* decisions. All outcomes of the decision are excluded except the *Application + Middleware Components* outcome that is included by all of the four MTLs. The relations between outcomes of the *Select Multi-Tenancy Level* and the *Select Migration Type* decision are an exact reflection of the inverse relations and will not be further explained, see Section 4.1.4 and Table A.14 respectively.

Define Scalability Level and Select Cloud Service Model: The outcomes of the *Select Multi-Tenancy Level* decision denote on which level an application shall *not* be shared. *Shared Middleware Multi-Tenancy* for example means that every tenant is provided with a dedicated application instance, but shares the middleware and all underlying hardware with others. In order to achieve the desired benefits of multi-tenancy as stated in Section 3.3, those underlying resources must be able to scale. Accordingly, the scaling level has to correspond at least to the level on which multi-tenancy is applied. Thus, it can be concluded that MTLs have an almost symbiotic relationship towards the scaling level.

In matters of the *Select Cloud Service Model* decision, a chosen MTL requires a service model that supports the separation of tenants on the desired level. That means that the cloud consumer must have control over those parts of the application leading to certain possible combinations, see Fig. 3.7 in Section 3.3 for reference. In summary, this results in the following relations, see Table 4.7, between the four MTLs and the outcomes of the *Define Scalability Level* and *Select Cloud Service Model* decision respectively:

- **Shared Application Multi-Tenancy:** If the application instance and everything below is shared the whole application stack has to be scaled. Thus, only the outcome *VM + Middleware + Application Level Scaling* is allowed and all other outcomes are excluded. An including relationship is not applicable because the scaling could also be taken over by the vendor. *Shared Application Multi-Tenancy* implies that the cloud consumer does not need any control of the separation of tenants since everything is shared. This means that any outcome under the *Select Cloud Service Model* decision is possible. For instance,

Table 4.7.: Subset of outcome relations of *Select Multi-Tenancy Level*.

Select Multi-Tenancy Level	Define Scalability Level								Select Cloud Service Model						
	No Scaling	*VM Level Scaling*	*Middleware Level Scaling*	*Application Level Scaling*	*VM + Middleware Level Scaling*	*VM + Application Level Scaling*	*Middleware + Application Level Scaling*	*VM + Middleware + Application Level Scaling*	*IaaS*	*PaaS*	*SaaS*	*IaaS + PaaS*	*IaaS + SaaS*	*PaaS + SaaS*	*IaaS + PaaS + SaaS*
MTL 0 (Hardware)	ex	a	ex	ex	a	a	ex	a	in	ex	ex	ex	ex	ex	ex
MTL 1 (OS)	ex	a	ex	ex	a	a	ex	a	in	ex	ex	ex	ex	ex	ex
MTL 2 (Middleware)	ex	ex	ex	ex	a	ex	ex	a	a	a	ex	a	ex	ex	ex
MTL 3 (Application)	ex	ex	ex	ex	ex	ex	ex	a	a	a	a	a	a	a	a

in the case of *IaaS* as chosen outcome, the consumer has full control and thus has to implement the sharing of the application stack from the VM upward individually [10], [13]. In contrast, if *SaaS* is chosen it is highly likely that the application stack can be assumed to be completely shared even though this is difficult to be verified in practice. However, due to the obfuscation by cloud providers regarding their underlying systems and implementation, an evaluation of isolation and sharing is almost impossible but nevertheless can be assumed [13].

- **Shared Middleware Multi-Tenancy:** If the middleware is shared, the minimum level of scaling is the *VM + Middleware Level Scaling* outcome. However, it is not prohibited to scale the application instance per tenant as well. Therefore, two allowing relationships have been defined whereas the rest of the outcomes are excluded. Allowed outcomes regarding the service model are defined towards *IaaS*, *PaaS* and *IaaS + PaaS* since any combination with the outcome *SaaS* would potentially share the application instance as well.

- **Shared OS Multi-Tenancy:** If the OS and through the implied one-to-one relation the corresponding VM are shared, the minimum scaling level is *VM Level Scaling*. As before, further scaling is possible leading to a total of four allowing relationships towards the scaling level outcomes that include the *VM Level Scaling* outcome. With respect to the service model, any outcome other than *IaaS* could not guarantee a separation of tenants above the VM level, thus an

including relationship has been defined, see Table 4.7.

- **Shared Hardware Multi-Tenancy:** On the lowest MTL level the same relations as for the *Shared OS Multi-Tenancy* apply since at least an IaaS solution is used.

4.4. Select Service Provider / Offering

All identified relations in the following are stated in detail in the appendix from Table A.16 to Table A.21.

4.4.1. Select Cloud Deployment Model

Define Cloud Hosting: Cloud deployment models and possible hosting alternatives have a straightforward relation, see Table 4.8. The outcome *Public Cloud* inherently uses *Off-Premise Hosting*, thus an including relationship has been identified [10]. Consequently, excluding relations exist towards all other outcomes. A *Private Cloud* can either be hosted on- or off-premise, thus two allowing relations have been identified. A combination denoted by *Hybrid Hosting* is not applicable because only one private cloud is considered and a separation would result in two independent private clouds hosted independently leading to a hybrid cloud scenario [10]. Nevertheless, that combination cannot be depicted through the definition of the outcome *Hybrid Cloud*, see Section 3.4.

Although in [10] community clouds are distinguished as onsite (on-premise) and outsourced (off-premise, operated by a third party) community clouds, they always entail some sort of hybrid hosting. Community clouds imply that various organizations are accessing and/or providing services. Thus, a complete on-premise or off-premise hosting would be questionable. To that end, the outcome *Community Cloud* includes *Hybrid Hosting* and excludes the other hosting outcomes. A hybrid cloud naturally entails an off-premise portion through the contained public cloud, hence two allowing relationship towards *Off-Premise Hosting* and *Hybrid Hosting* have been defined. Towards *On-Premise Hosting* an excluding relation has been stated.

Define Roles of Responsibility: *Public Cloud* as well as *Private Cloud* depict a deployment model that do not allow any form of combinatorial roles because at least two cloud environments would be needed. Through the stated assumptions regarding the *Define Roles of Responsibility* decision in Section 3.4, the following relations

Table 4.8.: Subset of outcome relations of *Select Cloud Deployment Model*.

Select Cloud Deployment Model	Define Roles of Responsibility							Define Cloud Hosting		
	Inhouse	*Managmenet*	*Outsourced*	*Inhouse + Management*	*Inhouse + Outsourced*	*Management + Outsourced*	*Inhouse + Management + Outsourced*	*On-Premise Hosting*	*Off-Premise Hosting*	*Hybrid Hosting*
Public Cloud	ex	a	a	ex	ex	ex	ex	ex	in	ex
Private Cloud	a	a	a	ex	ex	ex	ex	a	a	ex
Community Cloud	ex	ex	ex	ex	in	ex	ex	ex	ex	a
Hybrid Cloud	ex	a	a	a	a	a	a	ex	a	a

can be inferred. A *Public Cloud* is, per se, not on-premise, thus only the outcomes *Management* or *Outsourced* apply, excluding all other outcomes. For the *Private Cloud* outcome the exact opposite applies, see Table 4.8.

The outcome *Community Cloud* cannot allow any combination including *Management* because it would mean that some off-premise third-party cloud resources have to be managed. Regarding a community cloud every organization may provide some services *Inhouse* and/or only consume them from someone else (*Outsourced*). In both cases, the contributing organizations and the hybrid hosting nature of a community cloud always lead to a combination, hence an including relationship towards *Inhouse* + *Outsourced* has been defined. All other outcomes are excluded.

A *Hybrid Cloud* excludes the *Inhouse* option through the implied public cloud. All other outcomes are allowed. For example an off-premise private cloud and a public cloud could either use *Management*, *Outsourced* or *Management* + *Outsourced*. However, the expressiveness of the *Hybrid Cloud* is limited since it does not distinguish specifically between a community or private cloud, see Section 3.4. Thus it cannot be directly indicated which combination has to be chosen.

4.4.2. Select Cloud Service Model

Select Application Components: All relations are identical to the relations defined in Section 4.1.3 for the other way around and will not be repeated.

Select Migration Type: Similar to the previous decision, the relations between *Select Cloud Service Model* and *Select Migration Type* are an exact reflection apart from the including relationship between *IaaS* and *Migration Type III*. This relation has been substituted by an allowing relation since *IaaS* does not necessarily entail *Migration Type III*.

Define Scalability Level: All relations reflect the relations for the reverse, see Section 4.2.1 and Table A.17.

Select Scaling Type, Select Elasticity Automation Degree and Select Scaling Trigger: All relations from the outcomes of the *Select Cloud Service Model* decision towards the outcome of these three decisions exactly match those as in the respective reverse case. The relation will not be recapped and can be obtained from Section 4.2.2, Section 4.2.3, Section 4.2.4 and Table A.18.

Select Multi-Tenancy Level: All relations correspond to the relations that are defined for the reverse case from *Select Multi-Tenancy Level* towards *Select Cloud Service Model* defined in Section 4.3.1. The only deviations are the two including relations that have been changed to allowing relations, see Table A.18. Thus, the outcome *IaaS* has four allowing relationships towards all outcomes of the *Select Multi-Tenancy Level* decision because naturally any MTL can be provided based on IaaS.

4.4.3. Define Cloud Hosting

Select Cloud Deployment Model: The relations from the outcomes of *Define Cloud Hosting* towards *Select Cloud Deployment Model* are an exact reflection as in the reverse case, see Section 4.4.1. The only exception is that the including relation from *Public Cloud* towards *Off-Premise Hosting* has been substituted by an allowing outcome for the other way around, see Table A.19. If *Off-Premise Hosting* is chosen, in addition to *Public Cloud* the outcome *Private Cloud* is also a viable option.

Define Roles of Responsibility: *On-Premise Hosting* has an including relation towards *Inhouse* because is has been defined that in the case of on-premise hosting, everything is controlled by the cloud consumer, see Section 3.4. This leads to excluding relations towards all other outcomes of the *Define Roles of Responsibility* decision.

Table 4.9.: Subset of outcome relations of *Define Cloud Hosting*.

Define Cloud Hosting	Define Roles of Responsibility							Define Resource Location	
	Inhouse	*Managmenet*	*Outsourced*	*Inhouse + Management*	*Inhouse + Outsourced*	*Management + Outsourced*	*Inhouse + Management + Outsourced*	*Data In Same Jurisdiction*	*Data In Different Jurisdiction*
On-Premise Hosting	in	ex	ex	ex	ex	ex	ex	in	ex
Off-Premise Hosting	ex	a	a	ex	ex	a	ex	a	a
Hybrid Hosting	ex	ex	ex	a	a	ex	a	a	a

Off-Premise Hosting on the other hand excludes the *Inhouse* option and any combinatorial outcome including it but allows all other outcomes, see Table 4.9. The last outcome, *Hybrid Hosting*, allows any combination including the *Inhouse* option since some part will be on-premise leading to a total of three allowing and four excluding relations.

Define Resource Location: If *On-Premise Hosting* is chosen it can be definitely presumed that the *Data In Same Jurisdiction* outcome applies. Otherwise it would contradict the assumption that data stored at a company site falls under the same jurisdiction as the company site itself, something that can be taken for granted. Thus, an including relation between those two outcomes has been stated and an excluding relation towards the other outcome has been identified, see Table 4.9.

Off-Premise Hosting allows both outcomes under the assumption that a cloud provider exists compliant to the desired jurisdiction. The same applies for the *Hybrid Hosting* outcome. Logically, as a combination of the former two outcomes data will fall inevitably under the same jurisdiction and if desired, under a different jurisdiction through the off-premise part. Therefore two allowing relations towards *Data In Same Jurisdiction* and *Data In Different Jurisdiction* has been stated. In the case of *Hybrid Hosting* as the chosen outcome a subsequent selection of *Data In Different Jurisdiction* would not depict the fact that the outcome *Data In Same Jurisdiction* still applies. However, this is negligible because the whole point of the *Define Resource Location* decision is to raise the awareness about the data location and to indicate

the fact that data might fall under a *different* jurisdiction, see Section 3.4, something that is clearly satisfied by the defined relations.

4.4.4. Define Roles of Responsibility

Select Cloud Deployment Model: The relations are a reflection of the reverse case from *Select Cloud Deployment Model* towards *Define Roles of Responsibility*, see Section 4.4.1. the single deviation is the relation between *Inhouse + Outsourced* and *Community Cloud*. *Inhouse + Outsourced* is the only option to enable the *Community Cloud* outcome but it does naturally allow the outcome *Hybrid Cloud* as well. Therefore, the including relation has been substituted by an allowing relation.

Define Cloud Hosting: All relations between outcomes are a reflection of the relations as defined the other way around, see Section 4.4.3. Though, all influencing relations are substituted through the including relationship type based on the assumptions stated in Section 3.4. The *Inhouse* outcome requires *On-Premise Hosting* whereas any combination that includes it needs *Hybrid Hosting*. The remaining two basic outcomes, *Management* and *Outsourced* and their combination, always require *Off-Premise Hosting* leading only to including relationships, see Table A.20.

4.4.5. Select Cloud Vendor

The relationship types defined in Chapter 3 with regard to the cloud vendor (affecting and binding) cannot be directly further refined on the level of outcomes. As stated in Section 3.4, in order to precisely denote a relation the specific offered services for a cloud vendor must be captured. For instance, selecting a specific cloud vendor would imply whether a private cloud is offered or not. In the negative case, the vendor outcome would exclude the *Private Cloud* outcome. Through the coarse grained placeholder outcome *Evaluated Cloud Vendor*, these relations cannot yet be depicted. As a consequence, a refinement beyond a simple binding relation between outcomes or affecting relations, respectively, is not possible. Therefore, the relations between decisions have been transferred one-to-one to the level of outcomes. For instance, an affecting relation exists from the *Select Cloud Service Model* decision towards the *Select Cloud Vendor* decision, see Section 3.6. Hence, for each outcome of the *Select Cloud Service Model* decision an affecting relation has been defined towards the *Evaluated Cloud Vendor* outcome. This approach has been applied to all

respective relations and the resulting relations can be obtained from Table A.22 to Table A.24.

However, both relationship types still denote valuable information. Namely, which of the picked outcomes will affect the task to select an appropriate cloud vendor and, in the reverse case, it can be easily indicated which decisions might be subject to the cloud vendor.

4.4.6. Select Pricing Model

No relations on the level of decisions were identified in Chapter 3. As a consequence no relations between outcomes can be examined.

4.4.7. Define Resource Location

Define Cloud Hosting: Similar to the reverse case, *Data In Same Jurisdiction* allows any of the three outcomes under the *Define Cloud Hosting* decision. The outcome *Data In Different Jurisdiction* on the other hand excludes the possibility for an *On-Premise Hosting* and allows *Off-Premise Hosting* or *Hybrid Hosting*. The same reasoning for the latter outcome applies as stated in Section 4.4.3.

4.5. Summary of the Extension

The conducted elaboration of the relations between outcomes discussed in this chapter results in an extended version of the decision support framework. Five relationship types have been defined that vary greatly in their occurrences as can be seen in Table 4.10. The table lists the amount of outcome relationships per type per decision. A summary per outcome has been avoided because it is of less significance since a decision is always made on the decision level between outcomes. Therefore it is for example more important to know which decision might lead to numerous including and excluding relations in order to identify meaningful entry points into the decisions support framework. Besides, the total number of relations per outcome within one decision is logically always the same. In total 1570 relations have been identified. The numbers of relations per decision vary greatly from 0 to 280. Interestingly, only 30 including relationships have been discovered in contrast to 676 excluding relations. However, despite their low number the including relations

Table 4.10.: Quantification of outcome relations.

Decision	in	ex	a	aff	eb	Decision	in	ex	a	aff	eb
Select Application Layer	0	63	49	0	0	Select Application Tier	0	0	0	0	0
Select Application Components	0	122	118	0	0	Select Migration Type	5	64	67	0	0
Define Scalability Level	0	91	181	8	0	Select Scaling Type	1	19	25	3	0
Select Elasticity Automation Degree	2	29	59	5	0	Select Scaling Trigger	1	19	40	3	0
Select Multi-Tenancy Level	10	97	29	4	0						
Select Cloud Deployment Model	2	22	16	4	0	Select Cloud Service Model	0	98	147	7	0
Define Cloud Hosting	2	21	16	3	0	Define Roles of Responsibility	7	30	12	7	0
Select Cloud Vendor	0	0	0	0	50	Select Pricing Model	0	0	0	4	0
Define Resource Location	0	1	5	2	0						

	Including	Excluding	Allowing	Affecting	Binding	Total
Occurrences	**30**	**676**	**764**	**50**	**50**	**1570**

are highly distributed across eight decisions. Thus, several decisions might entail a specific outcome of another decision.

It is clearly visible in Table 4.10 that four out of the sixteen decisions play a significant role due to their number of excluding relations:

1. The *Select Application Components* decision limits tremendously, through its expressiveness regarding the amount and type of components, the possible migration types, scaling options and service models. Also a distinction between a complete and a partial migration can already be expressed further limiting the available choices especially with regard to the *Select Multi-Tenancy Level* decision.

2. Through the including relations from the *Select Migration Type* decision a selection of a migration type might determine the *Select Application Layer* as well as *Select Application Components* decision and might also lead to a required

service model. As a possible consequence the problem space is significantly reduced because a required service model will in turn, through the high amount of relations per outcome (36), further limit possible selections.

3. With regard to the *Select Multi-Tenancy Level*, the most including relations (10) exist. This is a natural consequence through the definition of the *Select Multi-Tenancy Level* decision and its implication of a complete application migration and required service model. If multi-tenancy is a fixed requirement at the very beginning of any migration decision, this decision would serve as an ideal entry point. However, if multi-tenancy is not a necessity a selection might actually be counterproductive and should be avoided since a partially occurring multi-tenancy for one component cannot be handled, see Section 4.3.1.

4. As discovered in Section 3.6.1 and Section 4.4, the *Select Cloud Service Model* decision is the only decision of its respective decision point that has influencing relations towards other decision points' decisions. The respective outcomes have in total 98 excluding relations (the second highest amount) thus, narrowing the choices particularly of the *Select Application Components*, *Define Scalability Level* and *Select Multi-Tenancy Level* decisions.

These four decisions follow the findings in Chapter 3 where their significance regarding their relations on the level of decisions has already been described. Yet, it clearly stands out that the *Define Scalability Level* decision has not been included above. Even though the *Define Scalability Level* decision has a high amount of excluding relations, several idiosyncrasies have to be considered. More specifically, through the *No Scaling* option numerous excluding relations exist within the decision point, see Section 4.2.1. Furthermore, the entire *Define Elasticity Strategy* decision point might be of specific interest to determine the requirements for a scaling strategy, yet it is unlikely that a migration decision would start with the determination of the preferred scaling options. Firstly, it is way more intuitive to select *what* has to be or can be scaled first (i.e. *Select Application Components*), therefore maybe already limiting the possible scaling options. Secondly, a predefined scaling strategy might not be necessary in the first place through the *Select Cloud Service Model* decision. In particular the service model can be deemed of higher relevance and as a more familiar choice to the decision maker in practice. Consequently, the *Define Scalability Level* decision might indeed serve as entry point into the framework but is not regarded as an essential decision to be considered from the very first start.

During the course of the extension several limitations have been discovered. In the following, respective examples are given to exemplify those restrictions:

- In the case of a migration that consists of multiple dependent single migrations, for example *Migration Type II* in combination with a *IaaS + SaaS* solution, the applicable restrictions cannot be denoted in detail for each contributing migration. For instance, *IaaS + SaaS* allows all scaling levels through the containing *IaaS* outcome. Yet, the scaling levels would not relate in any part to the *SaaS* portion that would be scaled by the provider. This partly bound relation cannot be reflected and also occurs in the reverse case, especially with regard to scaling decisions. An applicable workaround is to reenter the decision support framework twice by splitting up the decision into two *Migration Type I* migrations (with their respective components), with one using *IaaS* and the other *SaaS*.
- A similar limitation occurs with regard to the *Select Multi-Tenancy Level* decision. A migrated component might be provided in a multi-tenancy manner by the provider. However, this can only be partly reflected through the binding relationships from the outcome *Evaluated Cloud Vendor* towards the MTLs.
- The affecting and binding relations have by definition a rather supportive character to the decision maker to indicate relations with regard to the cloud vendor. Through the coarse grained *Evaluated Cloud Vendor* outcome, more detailed recommendations are not possible. A complete listing of all vendors and their offerings, however, is not feasible, see Section 3.4.
- Due to the stated assumption regarding the *Hybrid Cloud* outcome of the *Select Cloud Deployment Model* decision, special cloud deployment combinations such as an on- and off-premise private cloud or a community cloud in combination with an off-premise private cloud cannot be depicted. The same applies with respect to the *Define Roles of Responsibility* decision. A hybrid cloud does not denote whether a community or a private cloud is included thus it cannot be depicted what kind of roles are applicable. For instance, in case of a public and community cloud all outcomes including the *Inhouse* outcome would be excluded. In both stated cases a reentering into the decision support framework by splitting up the decision as described above can be applied.

The conclusion from the stated examples is that the framework is limited in its ability to holistically depict all influences in one large migration project, e.g., different scaling strategies per components, which parts are migrated with a specific service model or what kind of multi-tenancy levels might apply. However, some of those restrictions can be overcome by reentering the framework with a subset of the migration task at hand as described above. Despite these limitations, the extended decision support framework has a significantly increased expressiveness and the possibility of a far more granular view of the relations with respect to CloudDSF.

5. Implementation of the CloudDSF+ Prototype

In this chapter a prototypical implementation to visualize the extended decision support framework will be developed. As stated in Chapter 1, the existing CloudDSF Prototype should be enhanced to visualize the newly refined relations between outcomes. At first, the insufficiencies of the existing CloudDSF Prototype will be stated and at the same time, requirements will be inferred in an informal manner in Section 5.1. The architecture of the prototype, the used technologies and off-the-shelf components to be used are specified in Section 5.2. Finally, in Section 5.3, the actual implementation with the different visualizations, as well as their functionalities are described.

5.1. Insufficiencies of the CloudDSF Prototype

The extended decision support framework and thus the newly defined relations between outcomes cannot be depicted with the existing CloudDSF Prototype. The same applies to the different relationship types between decisions. This is due to the fact that all visualizations in the CloudDSF Prototype were developed for a static traversal through the knowledge base to either show the hierarchical nature of the knowledge base (tree, treemap and partition layout), the relations between decisions and tasks (force layout) or the relations among decisions and between decisions and tasks (chord layout) [46]. Therefore, the layouts are not appropriate to let a user interact with the knowledge base or to give feedback about the implications of a chosen decision. Of course, this was at the time of implementation neither in the scope nor possible due to the lack of relations between outcomes.

In order to enhance the functionality and support the visualization of the extended decision support framework, two major implementation areas have been identified.

Firstly, static visualizations similar in their purpose as the already implemented layouts are needed. They must show the complete knowledge base including the relations and the respective relationship types between outcomes or decisions. It must be possible to easily identify the existing relations and to focus on specific relationship types and/or parts of the knowledge base. Secondly, a dynamic, interactive visualization must be provided to enable a navigation through the knowledge base, i.e., depicting the impact of chosen outcomes towards other decisions' outcomes and informing the user about possible conflicts. This would be a major step towards better support for decision makers.

Currently, the knowledge base is encoded in a manually created JSON file that is also used for the visualizations of the CloudDSF Prototype. Even though the JSON file is not extremely large, an incorporation of the undertaken changes with respect to decisions and their outcomes is not easily possible since every record would have to be manually altered, added or removed. With respect to the updated decision support framework knowledge base, which has many more relations, a manual approach would be very cumbersome and error prone, hence not feasible anymore. Besides, the encoding of the knowledge base in a JSON file is an inappropriate representation to be read or edited by a user. Logically, a more user friendly encoding of the knowledge base and an automated generation of the necessary input files for the new and the existing CloudDSF Prototype visualizations must be provided to enable future enhancements.

5.2. Architecture and Technologies

In order to improve the existing CloudDSF Prototype, the newly developed Cloud Decision Support Framework Plus Prototype (CloudDSF+ Prototype) is based on [46] and has been forked from the respective code repository[1]. The source code is publicly available[2] at *GitHub* under the *Apache 2.0* license[3].

As a logical consequence of the fork, the architecture of the CloudDSF+ Prototype is similar to the architecture in [18] and is depicted in Fig. 5.1 including further needed components and resources. As it is indicated in the figure, the file representing the knowledge base, see Section 5.2.1, is automatically parsed, see Section 5.2.2, and then transformed into two JSON files that serve as resources for the CloudDSF+

[1] Source code of the CloudDSF Prototype: `https://github.com/adarsow/clouddsf`
[2] Source code of the CloudDSF+ Prototype: `https://github.com/bametz/clouddsfPlus`
[3] `http://www.apache.org/licenses/LICENSE-2.0`

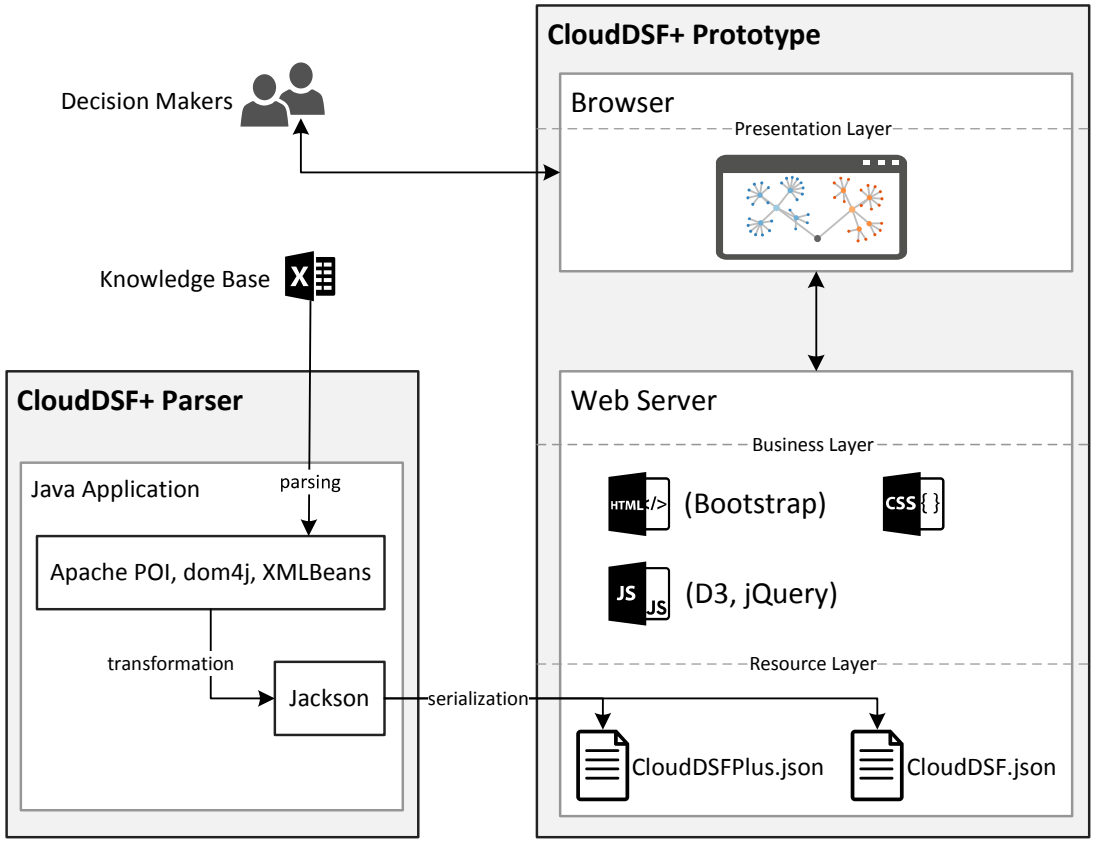

Figure 5.1.: Architecture and used technologies of the CloudDSF+ Prototype.

Prototype. The architecture of the CloudDSF+ Prototype is a simple static web application hosted in a web server to serve the web pages and necessary resources to the client's browser that visualizes the implemented layouts.

The amount of interaction between client and server is very low since all computations are executed on the client side after the initial request. Of course, the technologies on the web server side only indicate the type of documents being served to the browser that actually executes and leverages them. The used technologies are briefly stated in the following, omitting further explanations or references about them due to their ubiquity, the amount of easily available information and the target audience of this work. The choice of technologies have been predetermined by the CloudDSF Prototype and through the remaining tasks at hand:

- For the CloudDSF+ Prototype standard web technologies like Cascading Style Sheets (CSS), Hypertext Markup Language (HTML), JavaScript (JS), jQuery and the framework *Bootstrap* including several plugins have been used. The

visualizations have been implemented with the Data-Driven Documents (D3) library in conjunction with the mentioned technologies.

- For the persistence of the knowledge base *Microsoft Excel* has been used, and as input for the visualizations JavaScript Object Notation (JSON) files. The transformation between the two formats is performed by a Java application. The decision for Java was straightforward because sophisticated APIs, namely *Apache POI*, *dom4j*, *XMLBeans* and *Jackson* are available and significantly facilitate parsing and serialization.

5.2.1. The CloudDSF+ Knowledge Base

The extended decision support framework knowledge base has been encoded in one *Excel* file, a common way to textually represent information and matrices (relations). As desired, the representation is more comprehensible and changes can be more easily carried out than in the previous representation as a JSON file. The *Excel* file has the following sheets:

- **Knowledge Base:** Contains all decision points, decisions and outcomes as well as additional information such as abbreviations, classifications and descriptions. Besides the additional information for the visualization, it corresponds first and foremost to the updated knowledge base from Table 3.1. All other sheets use it for reference to enable a fast renaming, adding or removing of elements and ensure consistent data across the whole file.
- **Decision Level:** Depicts all relations between decisions (i.e. including, affecting, binding and "n/a" for not applicable). The reading order, valid for all sheets, is always from left to top to determine the direction of the relation.
- **Required Level:** Similar to the previous sheet but only depicts requiring relations between decisions.
- **Outcome Level:** Contains all relations on the level of outcomes. To depict the different relationship types the following abbreviations are used: allowing (a), including (in), excluding (ex), affecting (aff), binding (eb).
- **Task Level:** This sheet is for the support of the CloudDSF Prototype visualizations only. It includes the CloudDSF tasks and their relations towards decisions. It is very critical to note that the tasks and the relations to the refined decisions have been transferred one-to-one from [18] and *have not been updated or reviewed*. Due to the minor changes to the overall meaning and purposes of decisions, the relations are likely to be correct.

The knowledge base is independent from any visualization or implementation and its information can be extracted and preprocessed to serve different needs. The described generation of JSON files in the following is therefore only one possibility to leverage the data.

5.2.2. The CloudDSF+ Parser

The identified missing possibility to easily propagate changes between knowledge base and visualizations has been tackled with the Cloud Decision Support Framework Plus Parser (CloudDSF+ Parser). Similar to the CloudDSF+ Prototype, the source code has been made publicly available[4] under the *Apache 2.0* license. The parser automates the extraction of the data from the knowledge base and generates an appropriate input for the D3 visualizations. The CloudDSF+ Parser is strongly coupled to the knowledge base file that serves as input and does not support fallbacks in case the structure is significantly altered. An overview of the CloudDSF+ Parser's classes and their attributes and relations are visualized in the appendix in Fig. A.1.

Two independent parsing procedures have been written that differ in the amount of gathered and serialized information. One procedure generates a JSON file for the newly implemented visualizations and the other procedure for the CloudDSF Prototype (legacy) visualizations. The information contained in the latter JSON file differs from the original due to adaptations regarding the implementation, see Section 5.3.1, leading to a more compact and concise file. Both procedures use the same classes and work roughly as follows. First, the knowledge base sheet is parsed and for each entity a corresponding object is created including all its information. Subsequently, the rows of the remaining sheets are iterated and for each identified relation a corresponding relation object is created. In the last step objects are sorted for a more comprehensible output and then serialized into the respective JSON file. As a consequence, the knowledge base can be automatically provided in an appropriate format for the visualizations, tremendously facilitating future changes. Also, the backward compatibility for the CloudDSF visualizations is ensured.

[4]Source code of the CloudDSF+ Parser: `https://github.com/bametz/clouddsfPlusParser`

5.3. The CloudDSF+ Prototype

As previously noted, the CloudDSF+ Prototype is based on a project fork of the CloudDSF Prototype. However, due to technical issues, see Section 5.3.1, the complete front end has been redeveloped for a more extensive use of the *Bootstrap* framework. This enables a more responsive layout and compact implementation. In fact the only part remaining of the previous prototype is the implemented visualizations that have been incorporated under the navigation point *CloudDSF Visualizations*, see Section 5.3.1. The new overall look and feel, the navigation bar, as well as the possible available subnavigation can be seen in Fig. 5.2.

Besides two informational sites about the project that will not be further discussed, two main navigation points are provided corresponding to the previously identified areas for implementation, see Section 5.1. Under *KB Visualizer*, the layouts for a static view of the knowledge base are provided, whereas under *KB Navigator* the dynamic view has been implemented.

5.3.1. CloudDSF Visualizations

As discussed previously, the existing visualizations from [46] have been added under a separate navigation point. Besides the functional insufficiencies for the updated decision support framework discussed in Section 5.1, due to its prototypical character, several technical shortcomings have been identified that are stated in the following:

- **General Layout:** *Bootstrap* is not used consistently i.e. omitted grid-layout classes, no toggling of navigation and an unnecessary amount of nested divs as wrappers to enable CSS styles that could be substituted by built-in bootstrap classes. CSS with redundant, unnecessary or non-applicable statements in part because of prohibiting or not using the cascading nature of CSS. Unused HTML items and elements that are not visible.
- **Visualizations:** The content of the layouts' Scalable Vector Graphics (SVG) elements overflow the boundaries leading to cut offs. Labels are sometimes not displayed correctly and some scaling issues exist. Deselecting a highlighted object does not update the corresponding information in the panel.
- **JavaScript Programming:** No encapsulation or structuring of the JS code for the visualizations in modules. Instead, all layouts are implemented in one large interdependent JS file. The global namespace is polluted and many redundant

variables are assigned. A lot of code clones exist and adaptations for elements that are not visible or not existent are triggered. In addition, the same JSON file, including redundant data structures to avoid preprocessing, is unnecessarily requested once for each layout at the same time and in a synchronous instead of an asynchronous manner.

The general layout problems have been fixed or became obsolete through the redevelopment of the front end and only partly remain within some of the static visualizations. In order to smoothly incorporate the visualizations and panel into the new layout, the dropdown has been substituted by the new subnavigation bar. In addition, CSS styles have been adjusted to achieve a coherent look and feel, while simultaneously significantly reducing the amount of statements. Also, unnecessary JS statements have been tackled and minor jQuery fixes have been carried out. This lead to a significant amount of commented JS and CSS statements as well as removed HTML objects without losing any functionality.

The visualizations themselves remained mostly untouched since problems regarding their implementation would trigger an extensive refactoring due to the aforementioned highly interdependent nature of the implementation. The same applies to the JS problems regarding encapsulation and polluting of the global namespace. Nevertheless, a considerable amount of fixes have been carried out including the correct use of the treemap. Wherever necessary, adaptations and preprocessing steps have been added to support the newly generated and more compact, as well as concise, JSON file. Additionally, the JSON file is now only requested once instead of five times, yet, due to the given implementation restrictions, still in a synchronous manner. The tree layout has been omitted (not included in the navigation) since it has been substituted by the newly programmed hierarchical layout, see Section 5.3.2, that provides the same visualization but with greater functionality, responsiveness, encapsulation and better cross-browser support.

Through these adaptations it was possible to adapt the existing layouts to visualize the newly refined knowledge base. However, since it is not in the scope of this thesis, the layouts have not been optimized resulting in minor visualization errors such as misplaced labels. In addition, all relationships between decisions, regardless of their type, are visualized as influencing relations because a distinction is not supported. Also, all relations between tasks and decisions have been taken from [18] as-is, thus validity is not ensured.

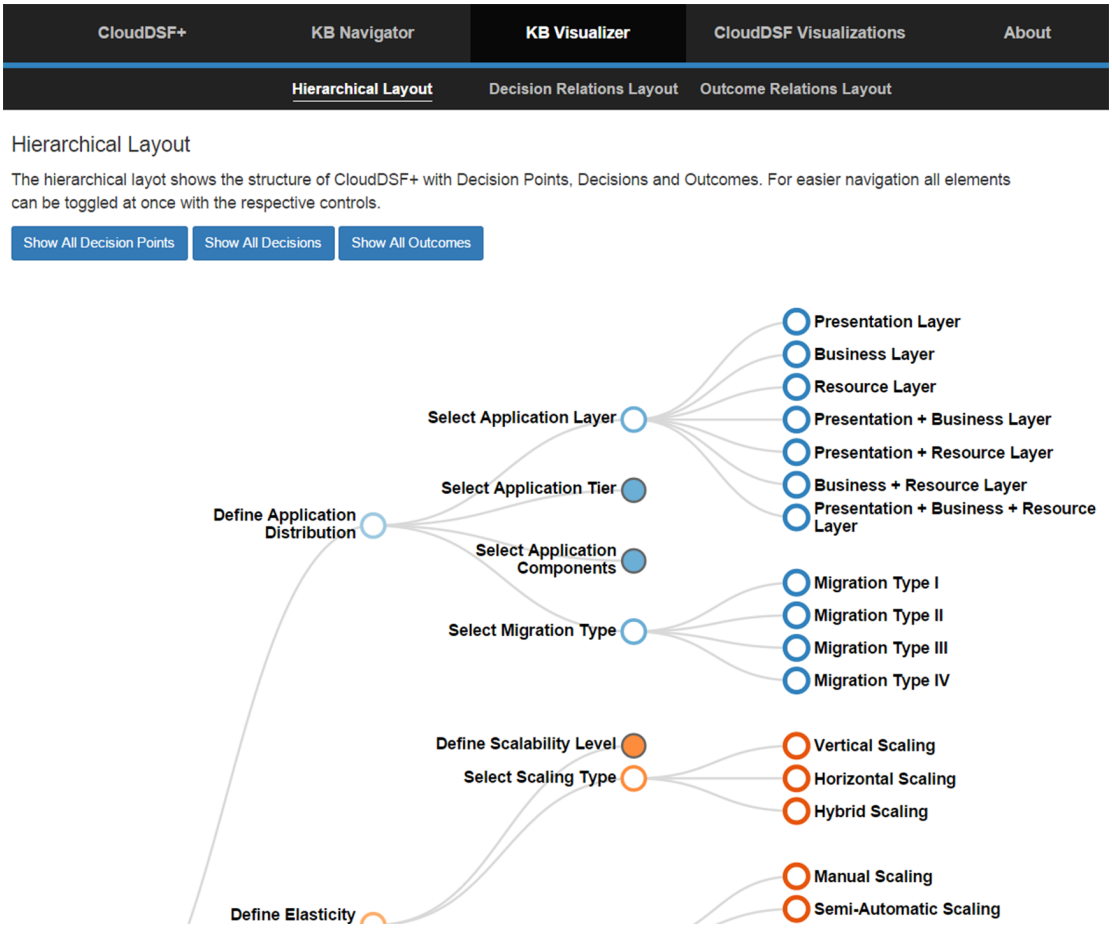

Figure 5.2.: Layout of the CloudDSF+ Prototype visualizing the knowledge base.

5.3.2. Knowledge Base Visualizer

Three new layouts have been developed visualizing the knowledge base. They either visualize the overall structure of the knowledge base (Hierarchical Layout), the relations between decisions (Decision Relations Layout) or the relations between outcomes (Outcome Relations Layout) in a static manner. The layouts use a global color scheme based on predefined values of a D3 color scale to paint the objects depending on the decision point and type to which they belong to. Abbreviations for the naming of elements are used wherever necessary to keep the visualizations clearly represented. The full name and explanation can always be obtained via an available tooltip that also shows a short description.

Figure 5.3.: Decision relations layout depicting the highlighted outgoing relations and connected decisions for the *Define Scalability Level* decision.

Hierarchical Layout: The layout shows a tree structure depicting the parent-child relations between decision points, decisions and outcomes in order to give an easy and fast overview of the knowledge base. In fact, this layout has already been used to generate the graphics in Section 3.1 to Section 3.4, depicting the updated outcomes of a decision. It is also partially visualized in Fig. 5.2. The layout is based on the same D3 layout as the tree layout from [46], yet it has been redeveloped and has not been adopted, see also Section 5.3.1.

Each node except those of the outcomes can be toggled to show or hide its immediate children. For convenience purposes three buttons have been added to collapse the tree to the level of all decision points, decisions or outcomes respectively. The layout automatically adapts to the size of the browser window to leverage wide screens. However, a fixed minimum width and height has been set to enable a decent view at all times.

Decision Relations Layout: The decision relations layout depicts all decisions clustered according to their decision point. This layout has been used to generate the figures in Section 3.5 to Section 3.6. Similar to the hierarchical layout, resizing is supported. Each of the four relationship types between decisions can be independently activated or deactivated (visible/hidden) to enable single or combined views of the relations. For usability purposes, all relationship types can also be toggled at once. In addition, each decision can be selected, highlighting all its outgoing relations while fading out all other relations and decisions as can be seen in Fig. 5.3. The relationship types are differently dashed and shaded to ease distinguishing them but they do not differ with respect to their color to avoid unnecessary confusion or any implication of good versus bad relationships. This also applies for the Outcome Relations Layout that will be discussed in the following.

Outcome Relations Layout: The third static view depicts all elements of the knowledge base grouped by their decision and/or decision point in a tree like structure, see Fig. 5.4. The functionalities are as follows:

- Any decision or decision point can be collapsed with a double click to reduce the amount of visualized objects and enable a more coarse grained view and/or focus on specific parts of the knowledge base. Relations towards outcomes that are not visible anymore are redirected to their respective decision or, in the case that those are collapsed as well, to their decision point, as can be seen in Fig. 5.4.

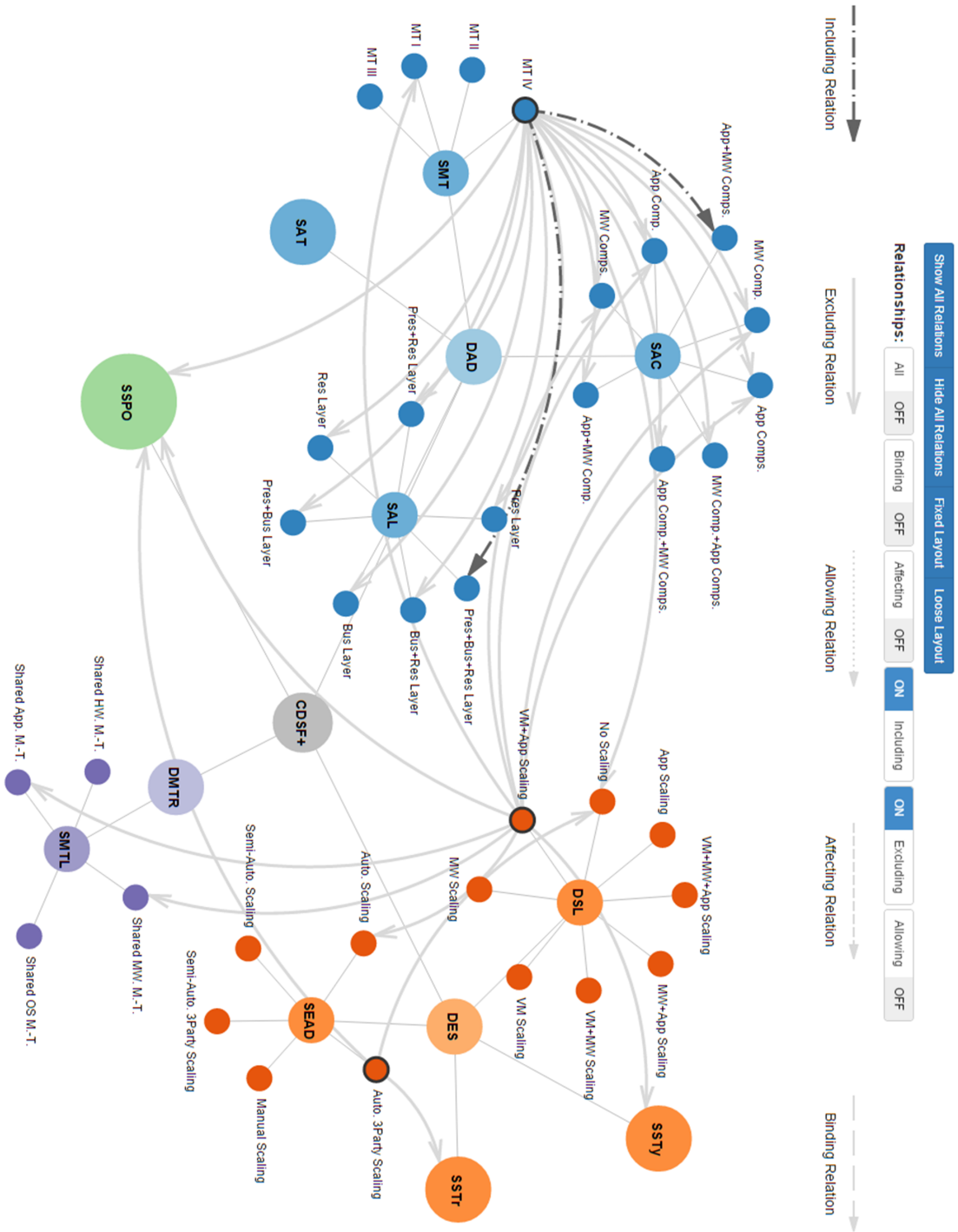

Figure 5.4.: Outcome relations layout with collapsed decisions, e.g., *Select Scaling Type* and collapsed *Select Service Provider / Offering* decision point.

- The layout can be fixed or loosened. In a fixed layout any node can be dragged towards any place while all other nodes stay in place. In a loosened layout, if one node is dragged all other nodes are rearranged based on the D3 layout parameters and simulation to enable a decent view at all times and a faster rearrangement.
- Every outcome can be clicked, visualizing or hiding its relations towards other outcomes.
- All relationship types can be activated/deactivated independently or together, to focus on specific relations in a clear and concise manner.
- The relations for all outcomes can be visualized or removed at once.

5.3.3. Knowledge Base Navigator

The dynamic layout, called *KB Navigator*, enables an interactive traversal through the knowledge base by visualizing the impacts of chosen outcomes. The provided functionalities are as follows:

- Decision outcomes are selectable and due to their XOR relation, a selection already excludes all other respective outcomes from further selections. The specified decisions as well as the excluded outcomes are appropriately marked (grayed out). However, the outcomes can still be selected in order to change a specified decision.
- Outcomes that are excluded by relations are faded out as well, whereas the included outcomes remain as-is. An automatic selection of included outcomes has been deemed inappropriate because this might create user confusion, thus the user has to select the included (necessary) outcome separately.
- In the event an excluded outcome is selected, a confirmation dialog appears asking whether the responsible excluding outcome(s) should be deselected and the desired outcome selected. This applies for both of the two previously described possibilities that lead to excluded outcomes.
- In the case a decision is determined de facto, i.e., only one outcome remains that can be selected, the decision is marked appropriately to indicate that fact. This applies similarly to decision points in case all of their respective decisions have been specified.

- By default, only the excluding and including relations for the last chosen outcome are visualized, as shown in Fig. 5.5. However, if desired, it is also possible to visualize the relations for all selected outcomes, see Fig. 5.6. For instance, it might be useful to show all relations to further examine why an outcome cannot be chosen. Afterwards, the default behavior can be reactivated to keep the view more clearly represented. Of course, the user can switch between these two behaviors at any time during the decision process.
- In the case an including and excluding relation are pointing towards the same outcome, a conflict exists and the outcome is highlighted appropriately, as demonstrated in Fig. 5.6. A selection of a conflicted outcome will show a dialog stating which outcomes are leading to the conflict. However, the choice to deselect the responsible outcomes automatically is intentionally not possible and has to be solved manually by the user.
- At any time, the user can toggle a visualization of requiring relationships between decisions to easily spot which decisions still have to be determined. The requiring relations are depicted as straight lines to clearly distinguish them from normal outcome relations. In addition, the lines as well as the contributing decisions are highlighted distinctively, see Fig. 5.5.

In addition, some general functions are provided whose respective buttons can be seen in Fig. 5.5. A selection can be saved or restored (export or import of a JSON file). The complete selection can be cleared, resetting all chosen outcomes to easily start a new selection. As stated above, only including and excluding relations are visualized since those are crucial for the potential restriction of outcomes. If desired, the remaining three relationship types can be toggled for the given and future selections at any time with the respective controls.

In conclusion, the *KB Navigator* provides an efficient means to guide decision makers through the knowledge base while giving them immediate feedback on how a chosen outcome will affect other decisions. Furthermore, selected, excluded as well as conflicting outcomes are distinctively illustrated, while relationships are only visualized on a need-to-know basis supporting a clearly represented view at all times.

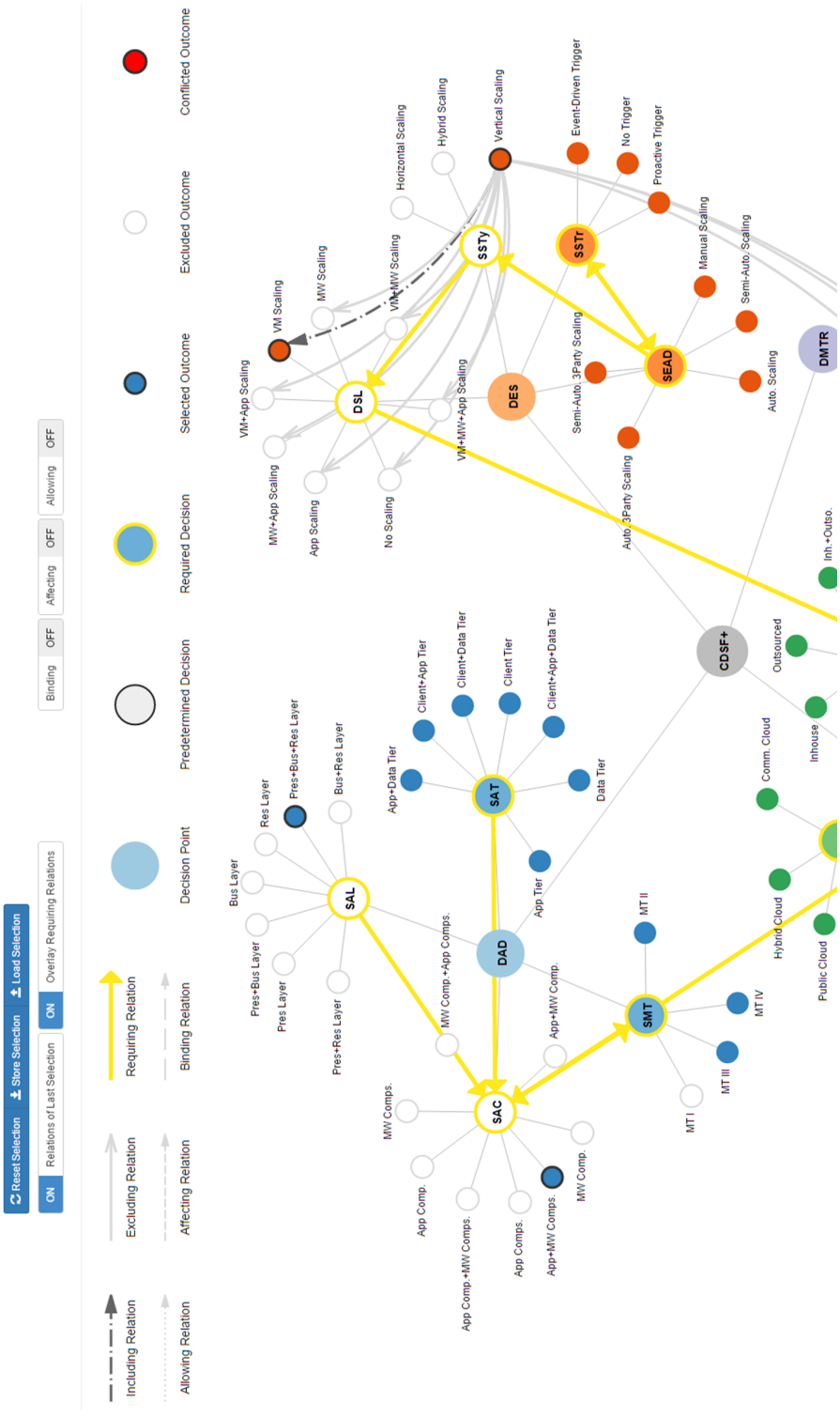

Figure 5.5.: Part of the *KB Navigator* depicting the toolbar, legend, requiring relations and the relations of the last specified outcome, *Vertical Scaling*.

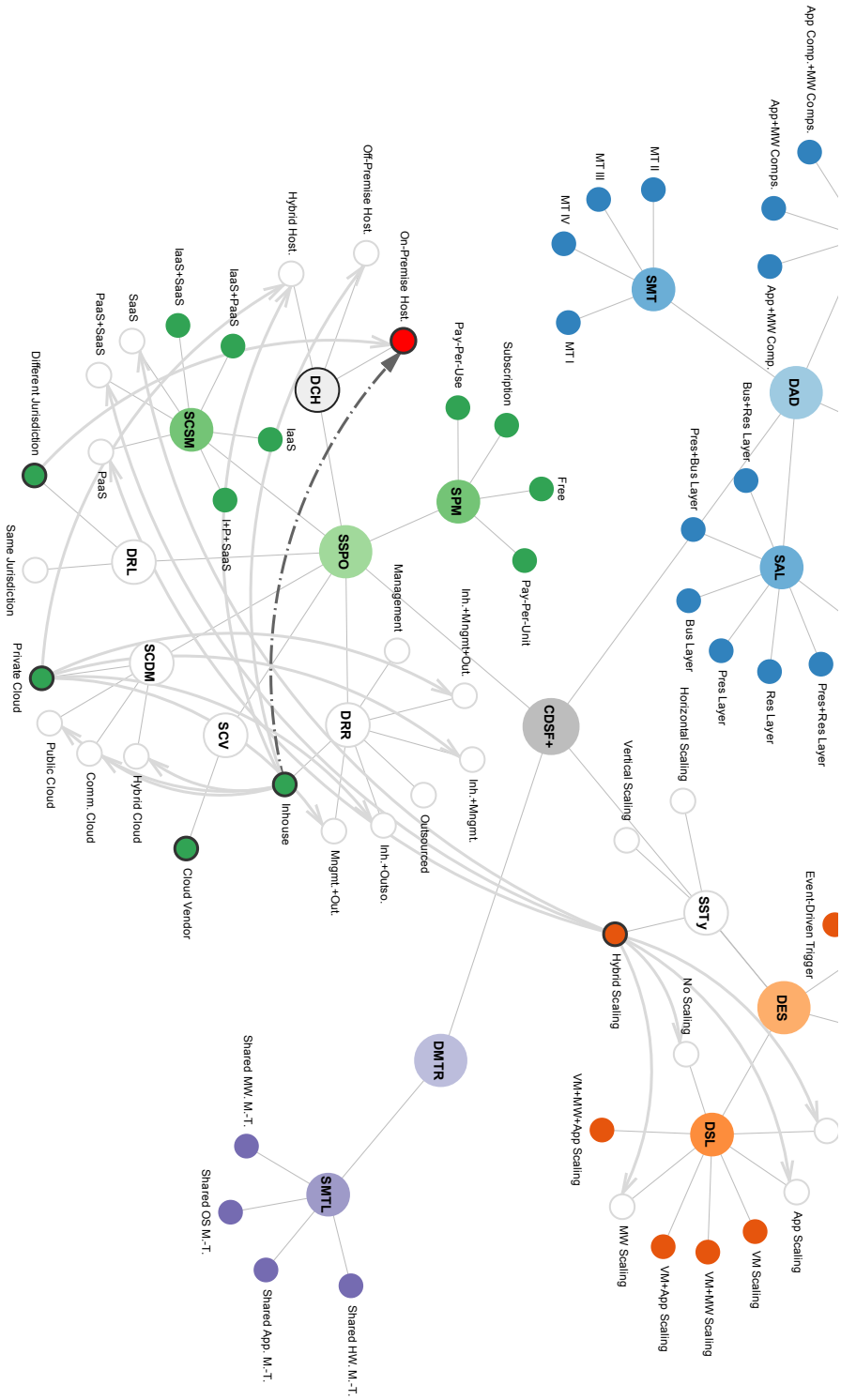

Figure 5.6.: Part of the *KB Navigator* showing the relations for all specified outcomes and a conflict at the *On-Premise Hosting* outcome.

6. Evaluation

The following chapter discussed the evaluation of the extended decision support framework. The evaluation is twofold. First, a validation regarding consistency and plausibility of the knowledge base is undertaken. Second, a use case from one of the related works discussed in Chapter 2 is used to demonstrate the efficacy of the knowledge base and the CloudDSF+ Prototype to support decision makers in migrating applications to the cloud.

6.1. Validation of the CloudDSF+ Knowledge Base

The validation of the knowledge base focuses on verifying its internal consistency. Before any validation can be carried out, the correct behavior of the CloudDSF+ Parser, as discussed in Section 5.2.2, has to be assured. For that purpose, a mock-up file has been created that corresponds exactly to the file described in Section 5.2.1 but is reduced in the amount of decision points and decisions, thus, also in relations to facilitate the determination of the expected output. The mock-up file has been parsed with the parser that has also been used to create the JSON file for the new visualization as described in Section 5.2.2. The resulting Java object has then been verified with a JUnit test comprising various assertions such as the amount and properties of decision points, decisions, outcomes, relations, relationship types etc. Therefore, it can be assured that the input is parsed correctly. Since the same parser is used for the knowledge base itself, this validity extends also to the actual knowledge base due to the identical file structure and used methods.

For the validation itself, eleven rules have been specified that must be satisfied by the knowledge base. The rules have been logically inferred based on the definitions and assumptions stated in Chapter 3 and Chapter 4 and are as follows:

1. On the level of decisions, only influencing, affecting, binding and requiring relationship types are allowed.
2. On the level of outcomes, only including, excluding, affecting, binding and allowing relationship types are allowed.

3. On the level of decisions only requiring relations can be combined with other relationship types. Therefore, influencing, allowing and binding relationships cannot coexist from one decision to another.
4. If a relation from decision A to decision B exists, there must also be relationships from any outcome of decision A to any of the outcomes of decision B.
5. If a relation from outcome A to outcome B exists, there must also be a relationship from the respective decision of outcome A to the respective decision of outcome B.
6. The relationship types on the outcome level have to correspond to the relationship types between the corresponding decisions. If on the decision level an affecting or binding relation exists, on the level of outcomes only the respective relationship types are allowed. In the case of an influencing relationship, only including, excluding or allowing relationships are allowed.
7. Binding and affecting relations are complimentary to each other. Logically, if a binding relation from decision A towards decision B exists, in the reverse case, an affecting relationship must be present and vice versa.
8. Rule 7, see above, also applies to the level of outcomes.
9. If an including/allowing relation from outcome A to outcome B exists, in the case a relation exists in reverse, it must also be of the including or allowing relationship type. Otherwise a contradiction would exist.
10. Any given outcome can only have one relation towards another outcome.
11. Between outcomes of the same decision an exclusive or relation were specified. Hence, as soon as an outcome is selected all others of the respective decision are not applicable anymore. Therefore, defined relations between outcomes of the same decision never apply and would unnecessarily pollute the knowledge base. As a consequence, any given outcome is only allowed to have relations towards outcomes of other decisions.

In order to check the stated rules, the CloudDSF object, see Fig. A.1, of the CloudDSF+ Parser has been extended with eleven methods, each of which correspond to a specified rule. Two of those methods can be seen in Listing 6.1. The verification methods have been tested with 16 JUnit tests to ensure their correct behavior and that errors are indeed captured as well as false positives avoided. Every test case gets an instance of the object representing the parsed mock-up file, i.e., the knowledge base reduction for which the satisfaction of all rules is assured, and follows the same schema. Firstly, the respective method is executed and checked if it is asserted

to true. Secondly, an error targeted by the corresponding tested method is intentionally added. Finally, the method is executed again but this time it is checked whether it is asserted to false to make sure that the induced error has been caught. By these means, the correct behavior of the implemented methods has been verified.

Listing 6.1: Verification methods for rule number 10 and 11.

```
/**
 * Check for every outcome relation if the decicions have relationship as well.
 *
 * @return
 */
public boolean checkDecRelForOutRel() {
  for (OutcomeRelation outRel : influencingOutcomes) {
    Decision decSource = getDecision(getOutcome(outRel.getSource()).getParent());
    Decision decTarget = getDecision(getOutcome(outRel.getTarget()).getParent());
    boolean found = false;
    for (DecisionRelation decRel : influencingDecisions) {
      if (decSource.getId() == decRel.getSource() && decTarget.getId() == decRel.getTarget()) {
        found = true;
        break;
      }
    }
    if (!found) {
      return false;
    }
  }
  return true;
}
/**
 * Checks if outcomes have a relation to themselves or towards outcome of the same decision.
 *
 * @return
 */
public boolean checkXOROutcomes() {
  for (DecisionPoint decisionPoint : decisionPoints) {
    for (Decision decision : decisionPoint.getDecisions()) {
      for (Outcome outcome : decision.getOutcomes()) {
        for (OutcomeRelation outRel : influencingOutcomes) {
          if (outcome.getId() == outRel.getSource()) {
            if (outcome.getId() == outRel.getTarget()) {
              System.out.println("Error: Outcome has a relation to itself");
              return false;
            } else if (outcome.getParent() == getOutcome(outRel.getTarget()).getParent()) {
              System.out.println("Fail: Outcome has relations towards outcome of same decision");
              return false;
            }
          }
        }
      }
    }
  }
  System.out.println("Success: All outcomes satisfy the XOR rule");
  return true;
}
```

In order to check the internal consistency of the entire knowledge base, the respec-

tive file discussed in Section 5.2.1 has been parsed and all verification methods have been executed on it. No errors occurred for any of the specified rules. Thus, it has been assured that the knowledge base is parsed by the CloudDSF+ Parser as expected and the defined rules are satisfied. Of course, this sanity check does not cover semantic errors such as a incorrectly defined outcome or decision relation as long as it is compliant to the specified rules. Nevertheless, the knowledge base is consistent and the defined relations are shown to be sound. Also, with regard to the visualizations, the correct behavior for the XOR relation between outcomes is ensured and that only one relation from one outcome to another exists consistently. In fact, the serialization of the knowledge base is aborted if any of the checks fail. With that mechanism in place, careless modifications of the knowledge base that would leave it in an inconsistent state are avoided and cannot be propagated to the visualizations inadvertently.

6.2. Efficacy of CloudDSF+

In Section 2.4 several frameworks for decision support for application migration to the cloud have been discussed, among them, the *Cloud Adoption Toolkit*. More specifically, in [31] a real world use case is included that will be used in the following to demonstrate the efficacy of the updated decisions support framework. At the outset, in Section 6.2.1, the available and necessary information about the use case is extracted. Subsequently, in Section 6.2.2 the dynamic view of the CloudDSF+ Prototype is used to derive migration strategies for the to be migrated systems.

6.2.1. Description of the Use Case

The use case in [31] deals with the information technology (IT) infrastructure of the *School of Computer Science* at the *University of St Andrews* that provides several services to their 60 members of staff and 340 students. Several of those services and their corresponding systems have been deemed suitable for a possible migration by the *Cloud Adoption Toolkit*. The predetermination of those systems does not interfere with the evaluation because the part of the extended decision support framework to be evaluated does not entail the selection of the systems to be migrated. In the following, the systems will be described based on the available information using the same naming convention as [31]:

1. **Archive:** Service that provides archive functionality to all of the storage services of the school with 560 Gigabyte of data. Originally hosted on a storage server.

2. **StaffRes and StudRes:** The *StaffRes* service enables staff to manage their teaching materials for courses/lectures. The *StudRes* service enables the procurement of a specified subset of these materials from the *StaffRes* service in a read-only manner by the students. Both services are predominantly used at the start and end of a term and thus have a bursty usage pattern. It can be assumed that both systems access the same resources and can be actually treated as a single application. Each service is hosted on an application server whereas the necessary data is hosted on a storage server.

3. **Website:** The school's website is outdated and a rebuild is considered to leverage cloud computing functionalities. In addition, the website suffers from performance problems that might occur due to excessive loads in the university network. The website is hosted on an application server.

4. **WebDev:** This service is used as a testing ground for the aforementioned website or as a backup in case the main server for the website is not available. This service is logically hosted in the same location as the website but is very rarely used.

5. **WebApps:** Under *WebApps*, services such as blogs, public wikis and software downloads are subsumed that are virtually hosted on a nondedicated *Apache* server because of their very small usage and resource consumption.

6. **Home directories mirror:** The home directories for all students and staff are mirrored with this service hosted on a storage server to provide a replica of their files.

7. **Teaching:** Student projects for various courses that involve technologies such as application servers or relational databases, thus requiring a server, can be hosted with the *Teaching* service. The service only runs for 24 weeks per year since it is only necessary during the terms.

In summary, the services are hosted on nine application and three storage servers within the university network. A simplified deployment model of the services to cloud resources from the original use case can be seen in Fig. 6.1. In the model, the terms virtual storage (e.g., éxtitAmazon S3 or *Amazon EBS* solution) or virtual machine (e.g., a *Amazon EC2* instance) are used to depict the necessary resources in an abstract manner [31]. At the time of publishing of the use case in 2012, based on the model, the authors calculated various pricing options for different migration

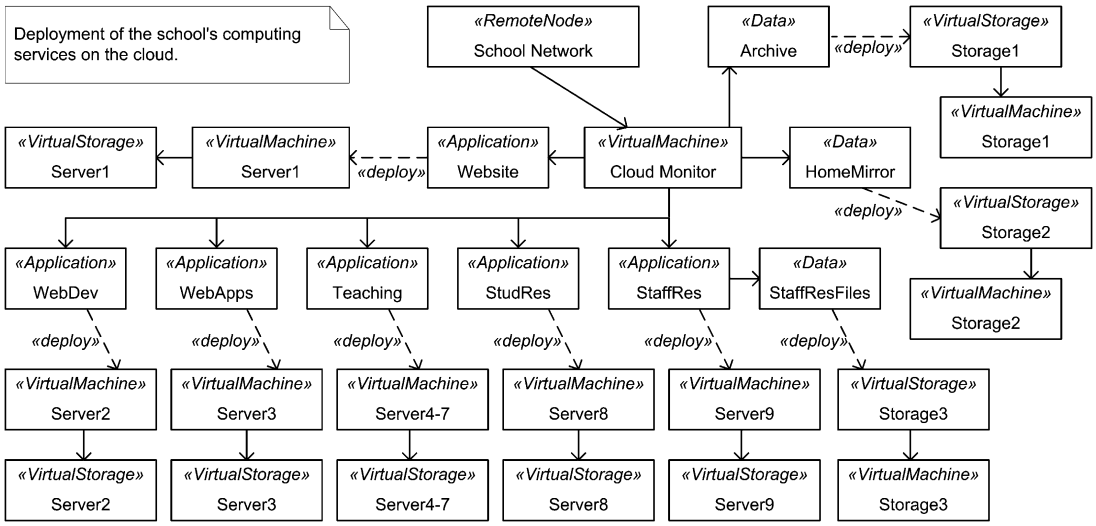

Figure 6.1.: Systems that are considered for migration [31].

options: 1. buying new hardware, 2. leasing the current equivalent amount of resources in the cloud, 3. leasing resources while leveraging the elasticity of the cloud, 4. synonymous to option three but with a 15% price increase and 5. also equivalent to three with a 15% price decrease in the next two years. The results were that leasing hardware while leveraging scalability was almost similar to buying new hardware. In the event of a price reduction, the cloud option became more attractive and monetarily advantageous.

It is critical to mention that the amount of price reduction in the last three years is far beyond the estimated 15% for two years. In fact, the four big cloud providers *Amazon*, *Google*, *Rackspace* and *Microsoft* in total slashed prices 22 times by an average of 23% in 2012 and 26 times by 26% in 2013 respectively [96]. The trend continued in 2014 through fierce competition among cloud providers and it is not predicted to stop anytime soon [97]. Of course, the price cuts are distributed across providers and their services. Nevertheless, the prices are decreasing sharply especially for computing and storage services, while at the same time, functionalities are being extended. Therefore, it can be assumed that the cloud computing option, hence leasing the resources, is the cheapest available option for the university. Furthermore, the university specifically considers to leverage the new possibilities of cloud computing and is under pressure from outdated server hardware.

6.2.2. Migration Strategies

In the following, a possible migration strategy is developed for each of the mentioned services. Due to some limitations regarding available information, logically inferred assumptions are made wherever necessary. Also, as exemplary cloud vendor *Amazon* is used, their available offerings known as *Amazon Web Services* will be referenced to exemplify the migration strategies[1]. Of course, any other cloud vendor would also be possible. The reason for applying *Amazon* is twofold. First, in the original use case *Amazon* was considered as the preferred choice of the university and all cost calculations have been done for *Amazon* cloud offerings. Second, *Amazon* can be considered as one of the most mature cloud vendors with very sophisticated solutions while being at the same time the price leader [96], [98].

Two main approaches to derive migration strategies are possible with respect to the updated decision support framework. Either all systems are treated as independent migration projects and the decision support framework is entered for each of them or, initially all systems are treated as one holistic application to decide decisions that apply to all of them and afterwards, the first approach is performed for a more detailed specification for each system while some decisions are already determined. The second approach has been deemed as more appropriate because the systems share the same specific domain, they have the same stakeholders, are very closely related to each other and the expected monetary benefits are similar. Due to the implementation of the *KB Navigator* this approach can be gracefully supported. First, the shared decisions are decided and the selection is stored as a file. Afterwards, for each system the stored file can be loaded and the remaining decisions specified.

Migration Decisions for all Systems: It is assumed that through the excessive load in the university network and the favorable pricing, see Section 6.2.1, as much as possible of the complete infrastructure should be migrated to the cloud. Therefore, *Off-Premise Hosting* has been chosen for all systems. This leads to several restrictions regarding the *Define Roles of Responsibility* and *Select Cloud Deployment Model* decisions. Also the jurisdiction must be specified, indicated by the requiring relations as can be seen in Fig. 6.2. Even though teaching materials might not contain sensitive data, files from the home directories as well as research/student projects, should not fall under a different jurisdiction. Furthermore, the domain of public education is normally restricted by data regulations thus *Data In Same Jurisdiction* has been selected. Also, all systems shall still be managed and controlled by the university's

[1] All information about *Amazon Web Services* referred to in the following can be obtained from `http://aws.amazon.com/products`.

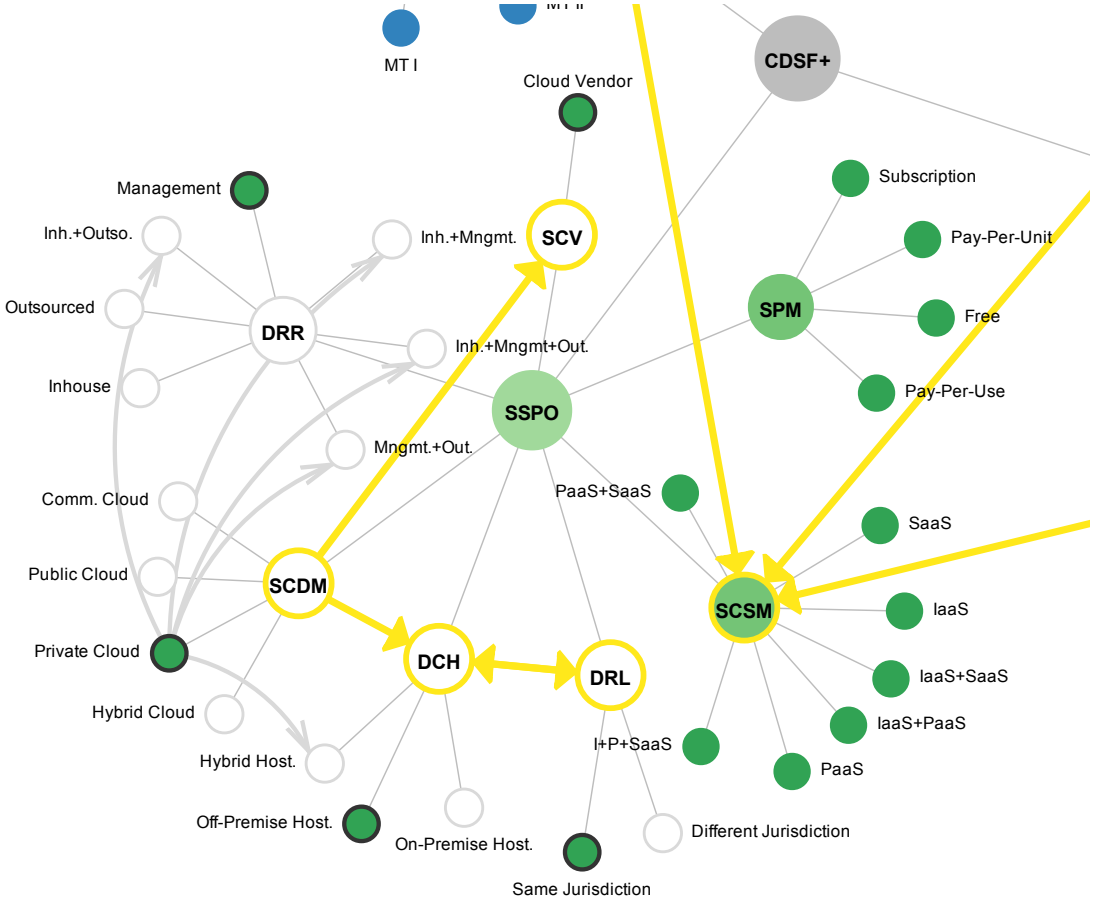

Figure 6.2.: Shared migration strategy for all systems of the use case.

administrators that support the users and are familiar with the functionalities. Logically, the role *Management* has been selected, thus leaving the burden of providing the hardware and its operation to the vendor.

The university is the only tenant and multi-tenancy is thus not required for any of the systems. Therefore, the *Define Multi-Tenancy Requirements* decision point will not be further considered in the following. With respect to the pricing model or the service model no general decision can be made. Due to the lack of a cost modeling component and the nonexisting relations, the *Select Pricing Model* decision will be specified to give an appropriate example, however, this must not represent the best solution.

Even though a specific cloud vendor cannot be selected, the outcome *Evaluated Cloud Vendor* has been specified and will be treated, as previously stated, as *Amazon*. However, due to the small amount of systems and only five full-time administrators, one

cloud vendor can be deemed as appropriate avoiding any unnecessary complexity or additional training. Besides the outcome *Community Cloud* for the *Select Cloud Deployment Model* decision, all three remaining outcomes are still selectable. Since all systems are currently hosted on-premise in a secured environment within the university network it can be assumed that only a *Private Cloud* satisfies the security requirements that have to be supported if the resources are hosted off-premise. Thus, the outcome *Private Cloud* has been chosen and will be used for all systems. The *Amazon Virtual Private Cloud*, for example, could be used to provide all systems within a secured perimeter enabling user authentication and secure access via a virtual private network. Consequently, five out of seven decisions have been specified for all systems, see Fig. 6.2. With respect to other decision points, at this point no decisions can be determined. The selection has been stored and will be used as a basis for the subsequent refinements of the migration strategies for the systems stated in Section 6.2.1.

Archive: The archive is a service used by all storage systems for the persistence of arbitrary data. The efficient scaling of the provided resources can be deemed as highly important for a storage solution in the cloud to avoid unnecessary costs for unused space, while at the same time always ensuring enough capacity. With regard to the *Define Elasticity Strategy* decision point this translates into the following decisions. *Automatic Third-Party Scaling* has been chosen to fully automate the scaling. In this case the third party would correspond to the cloud vendor *Amazon*. As a consequence, the *Select Scaling Trigger* decision became obsolete. As scaling level *Middleware Level Scaling* and as scaling type the remaining outcome *Hybrid Scaling* has been chosen. The reason that in the case of storage as a service solution the VMs are abstracted away. In the case of *Amazon S3*, for instance, buckets are provided that abstract the underlying infrastructure to store and retrieve data. Scaling of the middleware seems therefore more appropriate because not only is the storage, but also the middleware that is responsible to handle the read and write requests and their distribution, has to accommodate an increase or decrease in requests. However, in the case of *Amazon S3* and through the selected automation degree, the scaling tasks would be handled by the vendor anyway.

It can be assumed that the archive is not a stand alone application but rather a supportive system. This assumption is supported by the fact that in Fig. 6.1 it is only depicted as a hosted data solution without any application portion. Therefore, the important functionality is the storage of data thus the *Resource Layer* and logically *Middleware Components* have been chosen to be migrated. As a result, the *Select Migration Type* decision is predetermined and the remaining outcome *Migration Type*

II has been selected. Since cloud offerings that provide simple and cheap storage capabilities are subsumed under IaaS such as *Amazon S3*, *IaaS* has been chosen from the already limited amount of outcomes under the *Select Cloud Service Model* decision. For the pricing model *Pay-Per-Use* has been selected which is typically preferred in such circumstances to only pay the amount of resources that are actually used [10].

It must be mentioned that indeed, a *Migration Type III* migration would also have been possible. However, *Migration Type II* has been deemed more suitable since it is a service functionality which is migrated rather than an application. However, in this particular case *Migration Type II* could actually be considered as complete application migration even though it is not specifically expressed by the outcome.

Home Directory Mirror: Similar to the archive, the *Home Directory Mirror* service is of rather supportive nature and does not entail a full-blown application. In fact, there is no difference between those two services as they are also similarly depicted in Fig. 6.1 which is why the same migration strategy can be applied. In [31], both services have been, indeed, considered to be migrated to *Amazon S3*.

Website: For the website of the university a complete rebuild is considered to fully leverage cloud computing benefits. Naturally, *Migration Type IV* has been chosen that depicts this fact. This already limits the amount of selectable outcomes across various decisions and predetermines the *Select Application Layer* and *Select Application Components* decisions. For those two decisions, the remaining outcomes *Presentation + Business + Resource Layer* and *Application + Middleware Components* have been selected. Also, the remaining service model outcome *PaaS* has been chosen. A possible cloud solution would be, for example, the *AWS Elastic Beanstalk* platform.

These selections lead to several limitations of the available scaling options. To be more specific because of *Migration Type IV* and *PaaS* as the selected outcome under the *Select Cloud Service Model* decision. In fact, *Application Level Scaling* for the scaling level and *Horizontal Scaling* for the scaling type are the only remaining options, thus predetermining both decisions as can be seen in Fig. 6.3. Those options are indeed the possibilities that *AWS Elastic Beanstalk* offers because the provisioned deployment environment providing the middleware abstracts away the lower level leaving only the application scaling to the user[2]. Also, basic websites that are state-

[2] AWS Elastic Beanstalk shows some idiosyncrasies for a PaaS solution because of giving users access to the underlying VMs which are corresponding to *Amazon EC2* instances. However, a user does not necessarily have to deal with them thus the reasoning remains valid.

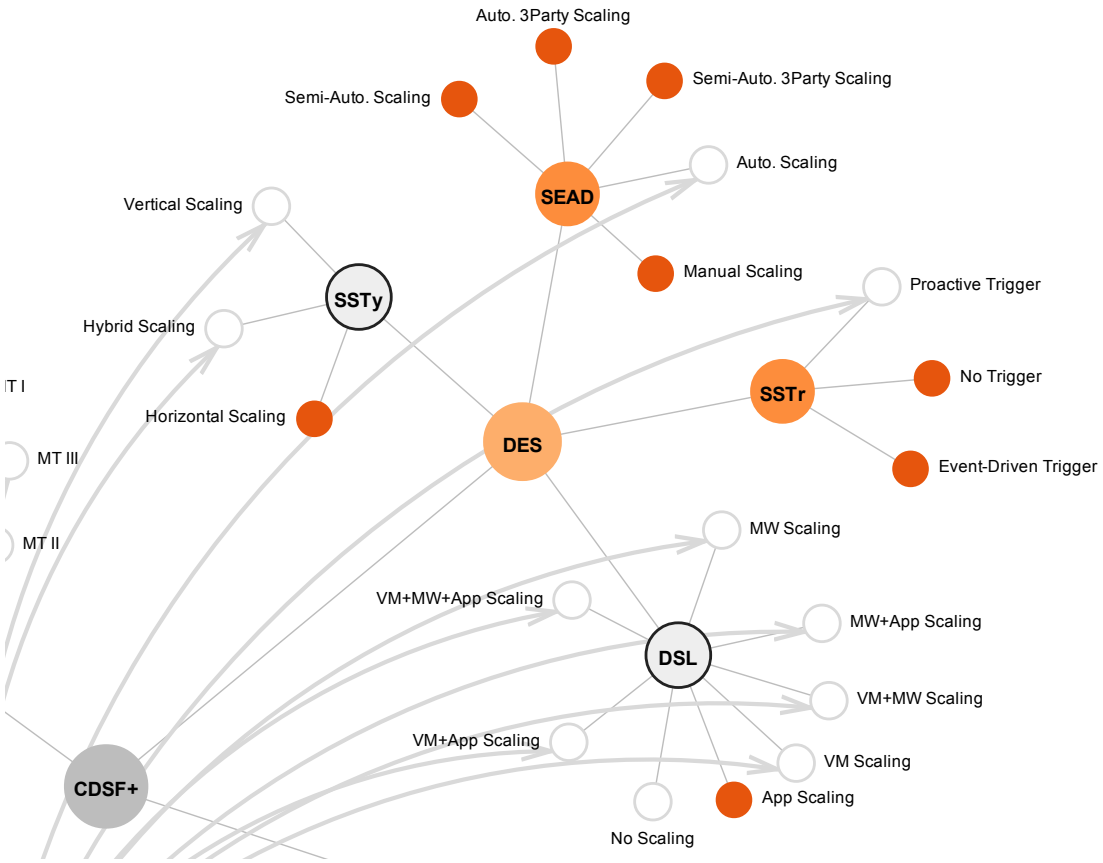

Figure 6.3.: Relations of the *PaaS* outcome towards the *Define Elasticity Strategy* decision point restricting as well as predetermining decisions.

less are normally scaled horizontally. For the automation, a *Semi-Automatic Scaling* has been deemed sufficient. The workload is easily predictable and peaks that might occur at the end or start of a term can be handled with an *Event-Driven Trigger* that can trigger the setup of another instance. Since the website has to be available all the time, a reserved instance with upfront payment can be deemed as the cheapest option since they grant the biggest discounts. Therefore *Charge-Per-Use (Subscription)* has been selected as pricing model.

WebDev: Cloud platforms such as the *AWS Elastic Beanstalk* platform are highly available and support failover with different availability zones. Therefore, a separately defined backup instance is rendered useless. Actually, it is highly likely that migrating the website to the cloud significantly increases its availability and performance compared to an on-premise hosting at the university. For development

purposes and testing of new updates that rarely occur [31], the existing instance can be easily cloned and instantiated on demand avoiding any unnecessary costs. Besides, by means of cloning the running instance, the testing environment always corresponds precisely to the productive environment inhibiting a common error-prone development chasm between the two. In conclusion, migrating the *WebDev* system is unnecessary which is why no migration strategy is to be elaborated.

Teaching The *Teaching* system is used by students for various projects that require server side technology. It seems therefore highly likely that specific configurations, adaptations and programs must be supported at any time. This requires the highest degree of control. Naturally, *IaaS* has been chosen in conjunction with *Migration Type III* for the migration of the whole application stack. This corresponds, for example, to a hosting on *Amazon EC2*. The undertaken selection predetermines the *Select Application Layer* and *Select Application Components* decisions whose outcomes have been chosen appropriately depicting the complete migration of the application.

For the scaling strategy, *VM Level Scaling* has been chosen. The server does not host one specific application but rather a variety of software development projects. Also, while servers are only needed during the term time it can be nevertheless assumed that during that time they have to be available without interruption. It seems therefore appropriate to use a simple *Manual Scaling* approach and consequently *No Trigger*. *Hybrid Scaling* has been selected as scaling type for two reasons. First, as depicted in Fig. 6.1 four servers are currently used for the service. Therefore, it is be possible that the demand varies throughout the term and over different years and to accommodate for this the option of for more or less instances might be useful. Second, a complete setup of a new instance especially during the term might be cumbersome. In that case, migrating to a bigger instance might be better to preserve already hosted projects or used endpoints by other systems. As pricing model *Pay-Per-Unit* has been selected, cheaper, reserved instances are not appropriate because they often entail a yearlong commitment.

StaffRes and StudRes: As previously discussed both systems can be treated uniformly. It can be assumed that they each provide a different view on an underlying database. It is not stated what kind of technologies are used, thus, two assumptions have been made. First, the data are stored in a relational database, and second the applications are web applications hosted on an application server. Both systems have the same predictable bursty usage pattern. More specifically, they are heavily used at the beginning and end of a term.

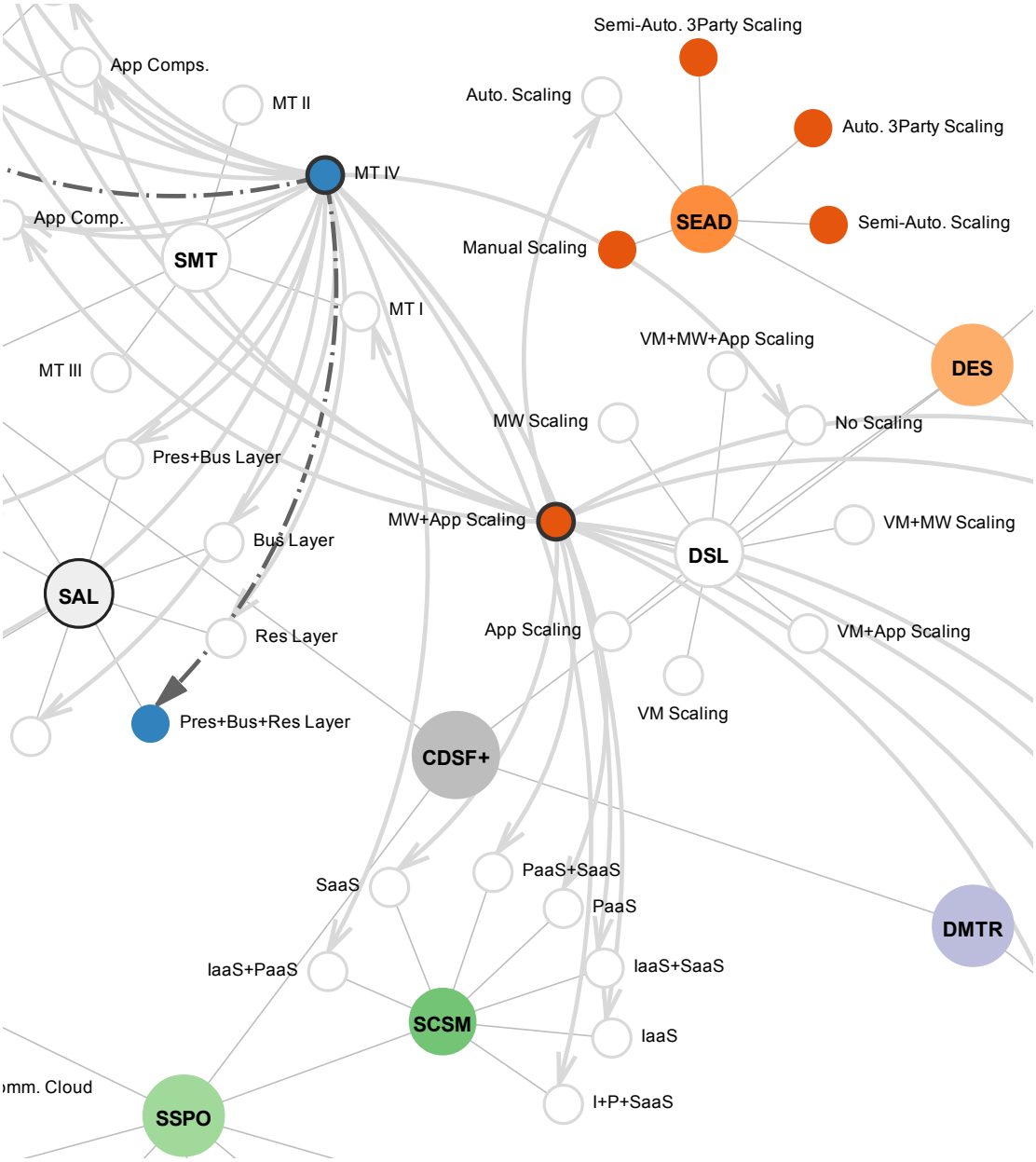

Figure 6.4.: Nonselectable service model (SCSM, lower middle) due to a conflicting selection of scaling level and migration type.

Seeing as the usage peaks are known in advance a *Semi-Automatic Scaling* in combination with an *Event-Driven Trigger* is sufficient. Since a middleware and two application front ends are migrated, the scaling should entail both parts. Hence, *Middleware + Application Level Scaling* has been chosen, intentionally leaving the underlying VMs out of scope in order to reduce the management burden. The re-

sult is that only *Horizontal Scaling* can be selected. This leads to a defined scaling strategy and to some restrictions with respect to other decisions.

It has been assumed that the application does not entail any specific technologies that might need a complete wrapping of the application as-is in a VM. Therefore, *Migration Type IV* has been selected determining the *Select Application Layer* and *Select Application Components* decision. In the course of this selection it became visible in the *KB Navigator* that this leads to no selectable outcomes for the *Select Cloud Service Model* decision, see Fig. 6.4. The reason was the selected scaling level would require an IaaS environment to support the scaling of the middleware. Indeed, scaling a database system is subsumed under middleware scaling and needs, as previously stated in Chapter 3, the scaling of the underlying resources. Two options are available to solve this problem. Either *VM Level Scaling* is included prohibiting *Migration Type IV* or the scaling level is reduced to *Application Level Scaling*.

The latter approach has been chosen using *PaaS* since the overall number of requests and queries to the database can be considered very low and easy to process thus does not require more sophisticated scaling options. Also, a PaaS based relational database solution such as *Amazon RDS* already automatically scales the amount of storage and satisfies the necessary scaling requirements for this scenario. The *Application Level Scaling* (instead of the previously defined *Middleware + Application Level Scaling*) logically applies only to the migrated front ends that can be hosted with, for example, *AWS Elastic Beanstalk*. All other previously selected outcomes remain as they are. For the pricing model either *Pay-Per-Use* or *Charge-Per-Use (Subscription)* might be reasonable.

WebApps: The migration strategy of the *WebApps* seems straightforward. Unfortunately it is not mentioned what kind of technology is used to persist the resources. Wikis as well as blogs usually rely on relational databases. However, in Fig. 6.1 only a simple storage for the VM is depicted that would be automatically handled in the case that the war files for the front end are migrated to the *AWS Elastic Beanstalk* platform. If it is assumed that a small relational database has to be migrated, that serves all services subsumed under *WebApps* as well, it would result in the same migration strategy as previously described for the *StaffRes* and *StudRes* services.

However, a different migration strategy has been chosen. To support the *WebApps* as-is including nonstandardized storage functionalities all of them are treated holistically in a *Migration Type III* and are migrated including all components as well as layers with *IaaS*, see Fig. 6.5. A suitable solution would be a small *Amazon EC2* in-

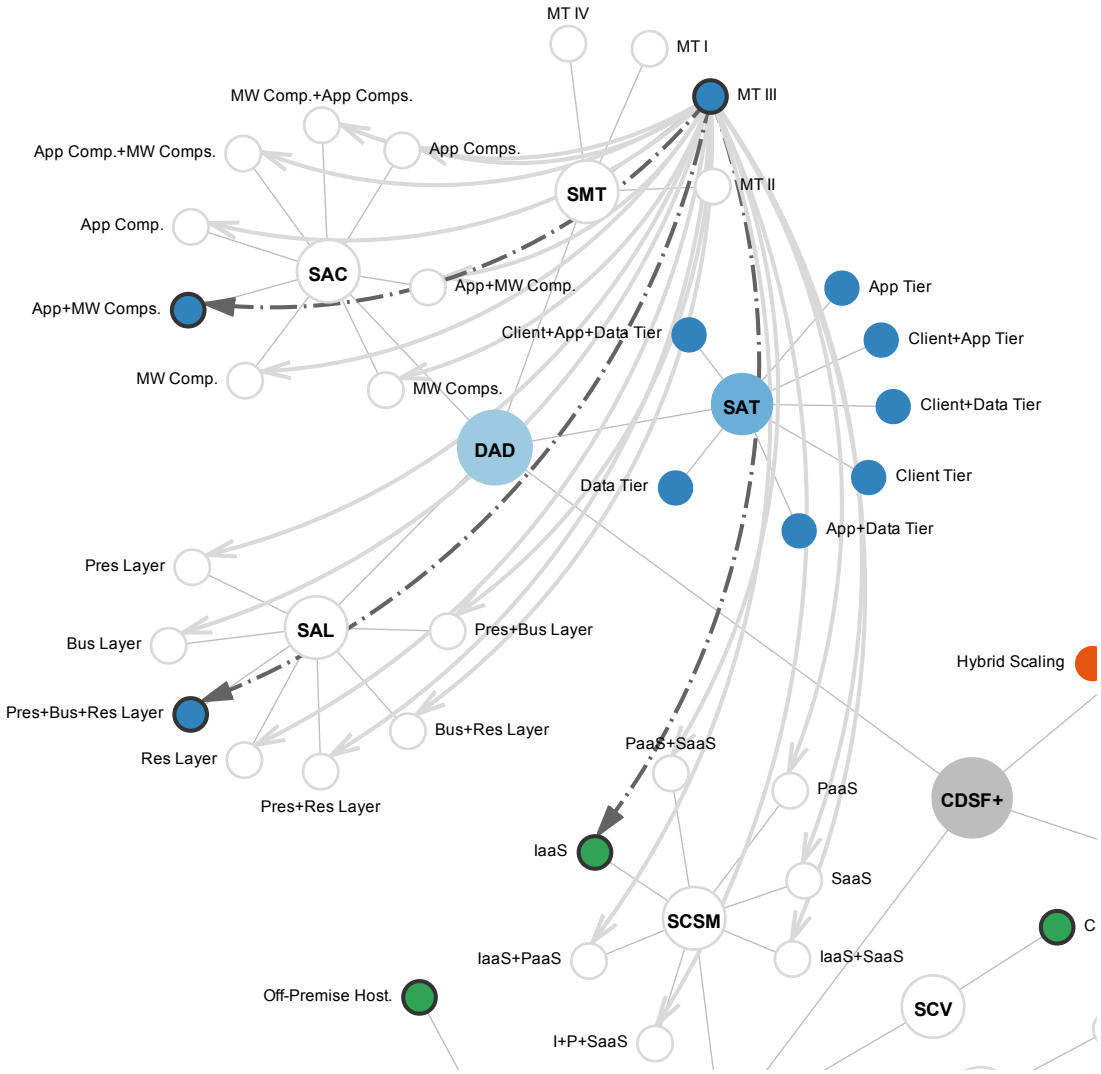

Figure 6.5.: Decisions for the application distribution of the *WebApps* service.

stance. Since they are rarely used, *No Scaling* has been chosen rendering any further decisions regarding the *Define Elasticity Strategy* decision point unnecessary.

6.3. Discussion

The undertaken validation in Section 6.1 guarantees that the encoded knowledge base actually satisfies certain specified rules. These rules have been inferred based on the assumptions and definitions stated during the refinement and extension of the

knowledge base. However, these assumptions and definitions must be evaluated as well. Therefore, a study similar to [18] with a wider range of participants, including the business domain, should be carried out in order to verify and validate their accuracy.

The applied use case shows that the CloudDSF+ Prototype serves its intended purpose and enables a fast, interactive way to derive consistent migration strategies for various scenarios. Conflicts, in terms of including and excluding relationships that simultaneously point to the same outcome are very rare due to a mechanism in place that deselects respective decisions if an excluded outcome is selected. The distinctive illustration of the current state of outcomes, decisions and decision points alike helps to guide decision makers smoothly through the framework. Also, a potential reentry into the framework has been demonstrated to split up a migration project into smaller subsets. In contrast to the *Cloud Adoption Toolkit* that developed a migration strategy solely based on costs, the CloudDSF+ Prototype covers far more aspects. Due to the scenario, the systems presented in the use case did not entail any form of multi-tenancy, which is why other use cases should be applied that cover that aspect.

The demonstrated versatility of the framework regarding decision chains, entry points or non-mandatory specifications can be clearly considered as an advantage. It can be reasonably argued that the absence of a more traditional, formal method of decision making actually complies quite closely with reality. As demonstrated by the use case, it might be ambiguous as to which kind of migration specifically should be performed making a distinction, for example of the migration type, difficult. Also, the possibility to initiate the decision making from different decisions might be valuable to accommodate different priorities for a migration such as high elasticity or required multi-tenancy.

The new requiring relations can be used to check whether all necessary relations have been specified. However, a mechanism is not yet in place to actively force decision makers to specify necessary decisions or guide them in a more formal manner to always guarantee a basic yet complete and sound migration strategy. Also, the limitations of the extended decision support framework that have been identified and discussed in Section 4.5, naturally, appear again. Due to the lack of specified cloud vendors, the reasoning with regards to possible cloud solutions for a migration strategy is currently left solely to the knowledge of the decision maker. The plethora of offerings demands a further refinement of the *Select Cloud Vendor* decision while at the same time keeping the framework concise. Analogously, this applies to the *Select Pricing Model* decision. For both decisions, an automated mechanism, e.g., via web services, to dynamically update the offers in order to accommodate the fast chang-

ing nature of cloud computing should be provided. Ultimately, this would enable the exploitation and further refinement of the binding and affecting relationships.

Currently, decision makers have to cope with the aforementioned ambiguities, also with regard to the application topology, by themselves. The fixed outcome relations intentionally prohibit selections that have been deemed impossible ensuring a consistent migration strategy. The underlying reasoning for the specified relations has been naturally based on definitions and logical assumptions. However, those are challenged by shifting definitions and/or borderline categorizations of characteristics and functionalities of cloud offerings in practice. This problem is further aggravated by the fast progress of cloud computing services and technologies that might enable combinations of outcomes that have been previously deemed as impossible. As a consequence, some migration scenarios might be supported in practice but the framework is not able to support the corresponding selection of outcomes. Hence, it might be inevitable to adapt the knowledge base in the future to accommodate those changes and further increase the expressiveness.

7. Conclusion and Further Research

The migration of legacy applications to cloud environments form a multi-dimensional problem with many interdependencies among and between technical as well as organizational aspects. As identified in Chapter 2, the available decision support approaches cannot be considered fully-fledged and further efforts have to be made. This thesis has been built upon the previously developed Cloud Decision Support Framework (CloudDSF) that aims to support decision makers to gather a sound information basis by means of tasks and, based on that, to guide them through necessary decisions that have to be specified prior to a migration. However, several deficiencies have been identified that have to be addressed to fully achieve this goal. One of these deficiencies, the missing relations between outcomes, has been addressed with this work.

To that end, the CloudDSF knowledge base has been refined in Chapter 3 entailing a review of the relations between decisions to ensure its appropriateness for an elaboration of the relations between outcomes. As a result, the knowledge base has been significantly altered while at the same time, by means of combinatorial outcomes for any possible combination of basic outcomes, the conciseness and expressiveness has been increased. Quantitatively this translates as follows: from the original 67 basic outcomes, 45 have been discarded and 38 new outcomes have been added. Overall, only 46% of all outcomes remained as-is whereas the remaining 54% of outcomes have been updated. The biggest changes, also in regard to their semantics, occurred with respect to the *Select Application Components* and *Define Scalability Level* decisions as well as to the *Define Multi-Tenancy Requirements* decision point.

In the subsequent refinement of relations, three new relationship types (requiring, affecting and binding) have been defined whereas one former relationship type (determining) became obsolete. Of the defined relations, 33 could not be confirmed or became deprecated and have been deleted while conversely 51 new relations have been added. The new requiring relationships between decisions can be utilized as a guide for decision makers whereas the new affecting and binding relationship types depict the relation of decisions towards or from a possible cloud vendor. Thus, in comparison to the original CloudDSF, it was possible to considerably enrich the information encoded at the decision relations level.

Based on this refinement, the extension of the decision support framework by elaboration of the relations between outcomes has been conducted. During the course of the elaboration, five relationship types (including, excluding, allowing, binding and affecting) have been specified that vary greatly in their occurrence. In total, 1570 relations have been defined ranging in their amount per decision between 0 and 280. The most important relations are the 30 including and the 676 excluding relations since those represent direct impacts between decisions. As a result of the extension, four decisions, namely *Select Application Components*, *Select Migration Type*, *Select Multi-Tenancy Level* and *Select Cloud Service Model* have been identified as highly important within the knowledge base. Their selection can significantly restrict and/or predetermine further possible choices and thus can be deemed as preferred entry points into the framework, however, this role has not been further investigated.

In order to enhance the existing CloudDSF Prototype to accommodate the extended decision support framework the new CloudDSF+ Prototype has been developed. Only the visualizations have been, with fixes, adjustments and the updated knowledge base, incorporated from the previous prototype whereas the web application has been redeveloped. In order to increase usability, the knowledge base has been encoded in a more suitable format. Furthermore, to facilitate future changes the knowledge base data has been automatically serialized with the new CloudDSF+ Parser into an appropriate format for the new as well as the legacy visualizations. Besides views to visualize the knowledge base, the relations between decisions, and the relations between outcomes, a dynamic view called *KB Navigator* has been implemented. It guides decision makers through the knowledge base while giving them immediate feedback on how a chosen outcome will affect other decisions in order to derive a migration strategy.

In the final step, an evaluation has been carried out comprising two parts. First, a validation has been performed. For that purpose, eleven rules have been specified that define a valid and consistent knowledge base. The rules are part of the implementation of the CloudDSF+ Parser whose correct behavior has been ensured. No inconsistencies have been found and a serialization of the data is prohibited in case any of the validation rules fail. However, possible rule compliant, yet semantically incorrect, defined relations have not been addressed with this validation. Second, a use case has been utilized to demonstrate the efficacy of the extended decision support framework by deriving various migration strategies for systems that differ in their nature and thus requirements.

During the extension and evaluation several shortcomings were identified. The extended decision support framework is limited in its ability to holistically depict all influences in one migration project, e.g., different scaling strategies per components,

which parts are migrated with a specific service model or what kind of multi-tenancy levels or pricing model might apply. However, some of these restrictions can be overcome by reentering the framework with a subset of the migration task at hand. Also, a mechanism is not yet in place that leverages the requiring relations to actively force decision makers to specify necessary decisions or guide them in a more formal manner to guarantee a simple yet complete and sound migration strategy.

Several deficiencies that were present prior to this work remain. Analogous to the decision points, an elaboration of the tasks has to be carried out. The relations between tasks and decisions as well as outcomes and vice versa have to be defined and a connection of the framework to a given application model must be possible. Also, a more comprehensive evaluation including the business domain should be carried out to verify the accuracy of the refined knowledge base. Additional still existing limitations are the non-refined *Select Pricing Model* and *Select Cloud Vendor* decisions. The *Select Cloud Vendor* decision does not enable detailed recommendations of cloud solutions or a further elaboration of the interdependencies beyond affecting and binding relationships. As a consequence, conclusions about possible cloud services for a migration strategy are solely left to the personal knowledge of the decision makers. Furthermore, due to the constantly varying cloud offers, a detailed elaboration of the outcomes of the *Select Cloud Vendor* decision and their relations which would lead to a tight coupling, seems inappropriate. A possible solution might be the extraction of generic cloud vendor requirements based on a defined migration strategy that are then mapped against the available offers. Also, the *Select Cloud Vendor* and *Select Pricing Model* decisions demand for an automatic mechanism, e.g., via web services to ensure up-to-date cloud services and pricing recommendations. However, extensive research is necessary to resolve these deficiencies.

During the course of the evaluation, one of the migration strategies advocated a new application rather than a migration. Even though the *Define Application Distribution* decision point is defined based on an existing topology it can, in fact, also depict a new, cloud native application. To that end, it is sufficient to remove the *Select Application Layer* and *Select Application Tier* decisions, *Migration Type I* and *Migration Type II* from the *Select Migration Type* decision and all outcomes except *Application + Middleware Components* from the *Select Application Components* decision. From this it follows that a complete application is considered through the single remaining outcome under *Select Application Components* and the two remaining migration types. The only additional change would be the removal of the *No Scaling* outcome since a cloud native application supports scaling. All other relations and outcomes can remain untouched and serve their intended purpose. A selection of *Migration Type III* would correspond to an application developed based on an IaaS environment, thus enabling full control over the scaling and multi-tenancy implementation. *Migration*

Type IV would correspond, for example, to an application deployment on a public PaaS offering. Hence, the extended decision support framework could be gracefully adapted for the decision support for engineering cloud native applications, thereby revealing a potential field of further research and significantly increasing its scope of application.

The demonstrated versatility of the extended framework regarding decision chains, entry points or non-mandatory specifications can be considered as an advantage. It can be reasonably argued that the absence of a more traditional, formal way of decision making actually complies quite closely with the reality. The distinctive depiction of the entities and their state of selection as well as the mechanism in place to resolve unnecessary conflicts supports decision makers smoothly through the decision process. In conclusion, it has been possible to show that the extended decision support framework and its implementation is indeed suitable to derive consistent migration strategies. Thus representing a considerable step towards more sophisticated decision support for application migration to the cloud.

Bibliography

[1] Gens, F. *IDC Predictions 2014: Battles for Dominance – and Survival – on the 3rd Platform*. IDC #244606. 2013. URL: http://www.idc.com/getdoc.jsp?containerId=244606 (visited on 08/23/2014) (cit. on p. 13).

[2] Gartner Inc. *Gartner Says Cloud Computing Will Become the Bulk of New IT Spend by 2016*. 2013. URL: http://www.gartner.com/newsroom/id/2613015 (visited on 08/23/2014) (cit. on pp. 13, 18).

[3] RightScale Inc. *State of the Cloud Report*. 2014. URL: http://www.rightscale.com/2014-cloud-report (visited on 09/06/2014) (cit. on pp. 13, 17, 18, 48).

[4] KPMG. *Technology Issues Notes: 2013 IT Spending Predictions Consensus*. 2013. URL: http://www.kpmg.com/be/en/issuesandinsights/articlespublications/pages/technology-issues-notes-feb-2013.aspx (visited on 08/23/2014) (cit. on p. 13).

[5] Armbrust, M. et al. "A view of cloud computing." In: *Communications of the ACM* 53 (4 2010), pp. 50–58. ISSN: 0001-0782. DOI: 10.1145/1721654.1721672 (cit. on pp. 13, 17, 18, 46).

[6] IBM. *Cloud computing insights from 110 implementation projects*. IBM. 2010. URL: http://www-935.ibm.com/services/us/leveragingit/learnings_from_100_early_cloud_adopters.pdf (visited on 08/23/2014) (cit. on pp. 13, 18).

[7] KPMG. *Modelling the Economic Impact of Cloud Computing*. 2012. URL: http://www.kpmg.com/AU/en/IssuesAndInsights/ArticlesPublications/Documents/modelling-economic-impact-cloud-computing.pdf (visited on 09/06/2014) (cit. on pp. 13, 17).

[8] Gartner Inc. *Gartner Executive Program Survey of More Than 2,000 CIOs Shows Digital Technologies Are Top Priorities in 2013*. 2013. URL: http://www.gartner.com/newsroom/id/2304615 (visited on 08/25/2014) (cit. on p. 13).

[9] Knorr, E. *9 trends for 2014 and beyond*. InfoWorld. 2013. URL: http://www.infoworld.com/t/cloud-computing/9-trends-2014-and-beyond-230099 (visited on 08/16/2014) (cit. on p. 13).

[10] Badger, L. et al. *Cloud computing synopsis and recommendations*. Vol. 800-146. NIST Special Publication. Gaithersburg, MD, USA: NIST, 2012. ISBN: 978-1-4776-2105-9. URL: http://csrc.nist.gov/publications/nistpubs/800-146/sp800-146.pdf (cit. on pp. 13, 16–19, 35, 36, 43–45, 53, 56, 58, 74, 84, 85, 118).

[11] Gartner Inc. *Forecast Overview: Public Cloud Services, Worldwide, 2011-2016, 2Q12 Update*. In collab. with Anderson, E. et al. Vol. G00234817. Market Analysis and Statistics. 2012 (cit. on p. 13).

[12] Gartner Inc. *Gartner Says By 2016, the Impact of Cloud and Emergence of Postmodern ERP Will Relegate Highly Customized ERP Systems to 'Legacy' Status*. 2014. URL: http://www.gartner.com/newsroom/id/2658415 (visited on 08/20/2014) (cit. on p. 13).

[13] Andrikopoulos, V. et al. "How to Adapt Applications for the Cloud Environment." In: *Computing* 95 (2013), pp. 493–535. ISSN: 1436-5057. DOI: 10.1007/s00607-012-0248-2 (cit. on pp. 13, 19, 27, 36, 41, 42, 55, 56, 59, 75, 76, 84).

[14] Vu, Q. H. and Asal, R. "Legacy Application Migration to the Cloud: Practicability and Methodology." In: *2012 IEEE Eighth World Congress on Services (SERVICES 2012)*. (Honolulu, HI, USA, June 24–29, 2012). Washington, DC, USA: IEEE, 2012, pp. 270–277. ISBN: 978-0-7695-4756-5. DOI: 10.1109/SERVICES.2012.47 (cit. on pp. 13, 17).

[15] Jamshidi, P., Ahmad, A., and Pahl, C. "Cloud Migration Research: A Systematic Review." In: *IEEE Transactions on Cloud Computing* 1 (2 2013), pp. 142–157. ISSN: 2168-7161. DOI: 10.1109/TCC.2013.10 (cit. on pp. 13, 19, 20).

[16] Andrikopoulos, V., Strauch, S., and Leymann, F. "Decision Support for Application Migration to the Cloud: Challenges and Vision." In: *Proceedings of the 3rd International Conference on Cloud Computing and Service Science (CLOSER 2013)*. (Aachen, Germany, May 8–10, 2013). SciTePress, 2013, pp. 149–155 (cit. on pp. 13, 20, 24, 25, 27, 28, 32).

[17] Andrikopoulos, V. et al. "CloudDSF – The Cloud Decision Support Framework for Application Migration." In: *Proceedings of the Third European Conference on Service-Oriented and Cloud Computing*. (Manchester, UK, Sept. 2–4, 2014). Ed. by Villari, M., Zimmermann, W., and Lau, K.-K. Vol. 8745. Lecture Notes in Computer Science. Springer, 2014, pp. 1–15. ISBN: 978-3-662-44878-6 (cit. on pp. 14, 16, 20, 21, 24, 25, 27, 29, 31).

[18] Darsow, A. “Decision Support for Application Migration to the Cloud.” Institute of Architecture of Application Systems. Master’s Thesis. University of Stuttgart, 2014. 157 pp. (cit. on pp. 14, 16, 19–21, 24, 27, 28, 31, 32, 34, 36, 37, 40, 46, 47, 49, 50, 52, 53, 55, 60, 67, 80, 81, 95, 97, 100, 124).

[19] Mell, P. and Grance, T. *The NIST Definition of Cloud Computing*. Vol. 800-145. NIST Special Publication. Gaithersburg, MD, USA: NIST, 2011. URL: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf (cit. on pp. 16, 78).

[20] Buyya, R., Broberg, J., and Goscinski, A. *Cloud computing. Principles and Paradigms*. Hoboken, NJ, USA: Wiley, 2011. ISBN: 978-0-470-88799-8 (cit. on pp. 16, 18).

[21] International Organisation for Standardization, International Electrotechnical Commission, and Institute of Electrical and Electronics Engineers. *Software Engineering – Software Life Cycle Processes – Maintenance: International standard ISO/IEC 14764 IEEE Std 14764-2006*. 2nd ed. New York, NY, USA: IEEE, 2006. ISBN: 0-7381-4961-6. DOI: 10.1109/IEEESTD.2006.235774 (cit. on p. 17).

[22] Suleiman, B. et al. “On understanding the economics and elasticity challenges of deploying business applications on public cloud infrastructure.” In: *Journal of Internet Services and Applications* 3 (2 2011), pp. 173–193. ISSN: 1867-4828. DOI: 10.1007/s13174-011-0050-y (cit. on pp. 17, 48).

[23] KPMG. *The Cloud – Changing the Business Ecosystem*. 2011. URL: http://www.kpmg.com/in/en/issuesandinsights/articlespublications/pages/thecloud-changingthebusinessecosystem.aspx (visited on 08/23/2014) (cit. on pp. 17–19, 46).

[24] Hugos, M. H. and Hulitzky, D. *Business in the Cloud: what every business needs to know about cloud computing*. New York, NY, USA: Wiley, 2011. ISBN: 978-0-470-61623-9 (cit. on pp. 17, 18, 46).

[25] Cloud Security Alliance. *The Notorious Nine: Cloud Computing Top Threats in 2013*. 2013. URL: https://downloads.cloudsecurityalliance.org/initiatives/top_threats/The_Notorious_Nine_Cloud_Computing_Top_Threats_in_2013.pdf (visited on 08/23/2014) (cit. on p. 18).

[26] Strauch, S. et al. “Decision Support for the Migration of the Application Database Layer to the Cloud.” In: *IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom 2013)*. (Bristol, UK, Dec. 2–5, 2013). Vol. 1. Piscataway, NJ, USA: IEEE Computer Society, 2013, pp. 639–646. ISBN: 978-0-7695-5095-4. DOI: 10.1109/CloudCom.2013.90 (cit. on p. 19).

[27] Burstein, F. and Holsapple, C. W. *Handbook on Decision Support Systems 1*. International Handbooks on Information Systems. Berlin, Germany: Springer, 2008. ISBN: 978-3-540-48712-8. DOI: 10.1007/978-3-540-48713-5 (cit. on p. 19).

[28] Power, D. J. *Decision Support Systems: Concepts and Resources for Managers*. Westport, CT, USA: Quorum Books, 2002. ISBN: 978-1-56720-497-1 (cit. on p. 19).

[29] Figueira, J., Greco, S., and Ehrgott, M. *Multiple Criteria Decision Analysis: State of the Art Surveys*. International Series in Operations Research & Management Science. New York, NY, USA: Springer, 2005. ISBN: 0-387-23081-5 (cit. on p. 19).

[30] Hajjat, M. et al. "Cloudward bound: Planning for Beneficial Migration of Enterprise Applications to the Cloud." In: *ACM SIGCOMM Computer Communication Review* 40 (4 2010), p. 243. ISSN: 0146-4833. DOI: 10.1145/1851275.1851212 (cit. on p. 21).

[31] Khajeh-Hosseini, A. et al. "The Cloud Adoption Toolkit: supporting cloud adoption decisions in the enterprise." In: *Software: Practice and Experience* 42 (4 2012), pp. 447–465. ISSN: 0038-0644. DOI: 10.1002/spe.1072 (cit. on pp. 21, 112–114, 118, 120).

[32] Beserra, P. V. et al. "Cloudstep: A step-by-step decision process to support legacy application migration to the cloud." In: *2012 IEEE 6th International Workshop on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA)*. (Trento, Italy, Sept. 24, 2012). Piscataway, NJ, USA: IEEE, 2012, pp. 7–16. ISBN: 978-1-4673-3001-5. DOI: 10.1109/MESOCA.2012.6392602 (cit. on p. 21).

[33] Menzel, M. and Ranjan, R. "CloudGenius: Decision Support for Web Server Cloud Migration." In: *Proceedings of the 21st International Conference on World Wide Web*. (Lyon, France, Apr. 16–20, 2012). Ed. by Mille, A. et al. New York, NY, USA: ACM, 2012, pp. 979–988. ISBN: 978-1-4503-1229-5. DOI: 10.1145/2187836.2187967 (cit. on p. 21).

[34] Menzel, M. et al. "CloudGenius: A Hybrid Decision Support Method for Automating the Migration of Web Application Clusters to Public Clouds." In: *IEEE Transactions on Computers* (2014), pp. 1–14. ISSN: 0018-9340. DOI: 10.1109/TC.2014.2317188 (cit. on pp. 20, 21).

[35] Chauhan, M. A. and Babar, M. A. “Towards Process Support for Migrating Applications to Cloud Computing.” In: *Proceedings of the 2012 International Conference on Cloud Computing and Service Computing (CSC)*. (Shanghai, China, Nov. 22–24, 2012). Los Alamitos, CA, USA: Conference Publishing Services, 2012, pp. 80–87. ISBN: 978-0-7695-4910-1. DOI: 10.1109/CSC.2012.20 (cit. on p. 21).

[36] Menychtas, A. et al. “ARTIST Methodology and Framework: A Novel Approach for the Migration of Legacy Software on the Cloud.” In: *2013 15th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*. (Timisoara, Romania, Sept. 23–26, 2013). Ed. by Björner, N. et al. Los Alamitos, CA, USA: IEEE, 2013, pp. 424–431. ISBN: 978-1-4799-3035-7. DOI: 10.1109/SYNASC.2013.62 (cit. on p. 21).

[37] Alonso, J. et al. “Cloud modernization assessment framework: Analyzing the impact of a potential migration to Cloud.” In: *2013 IEEE 7th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA)*. (Eindhoven, Netherlands, Sept. 23, 2013). Piscataway, NJ, USA: IEEE, 2013, pp. 64–73. ISBN: 978-1-4673-4889-8. DOI: 10.1109/MESOCA.2013.6632736 (cit. on pp. 21, 22).

[38] Frey, S. *Conformance Checking and Simulation-based Evolutionary Optimization for Deployment and Reconfiguration of Software in the Cloud*. Vol. 2014/1. Kiel Computer Science Series. Dissertation, Faculty of Engineering, Kiel University. Department of Computer Science, CAU Kiel, 2014. ISBN: 978-3-7322-9734-4 (cit. on pp. 21, 24).

[39] Juan-Verdejo, A. and Baars, H. “Decision support for partially moving applications to the cloud.” In: *HotTopiCS ’13: Proceedings of the 2013 International Workshop on Hot Topics in Cloud Services*. (Prague, Czech Republic, Apr. 21–24, 2013). Ed. by Kounev, S., Zschaler, S., and Sachs, K. New York, NY, USA: ACM, 2013, pp. 35–42. ISBN: 978-1-4503-2051-1. DOI: 10.1145/2462307.2462316 (cit. on pp. 21–23).

[40] Juan-Verdejo, A. et al. “Moving Business Intelligence to Cloud Environments.” In: *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. (Toronto, Canada, Apr. 27–May 2, 2014). Piscataway, NJ, USA: IEEE, 2014, pp. 43–48. ISBN: 978-1-4799-3088-3. DOI: 10.1109/INFCOMW.2014.6849166 (cit. on pp. 21, 23).

[41] Juan-Verdejo, A. et al. “InCLOUDer: A Formalised Decision Support Modelling Approach to Migrate Applications to Cloud Environments.” In: *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*. (Verona, Italy, Aug. 27–29, 2014). Ed. by Rabiser, R. and Torkar,

R. Piscataway, NJ, USA: Conference Publishing Services, 2014, pp. 467–474. ISBN: 978-1-4799-5794-1. DOI: 10.1109/SEAA.2014.55 (cit. on pp. 21, 23).

[42] Ahmad, A. and Babar, M. A. "A Framework for Architecture-driven Migration of Legacy Systems to Cloud-enabled Software." In: *WICSA '14 Companion: Proceedings of the WICSA 2014 Companion Volume*. (Sydney, Australia, Apr. 7–11, 2014). Ed. by Liu, A. New York, NY, USA: ACM, 2014, 7:1–7:8. ISBN: 978-1-4503-2523-3. DOI: 10.1145/2578128.2578232 (cit. on pp. 21–23).

[43] Abdelmaboud, A. et al. "A Comparative Evaluation of State-of-the-Art Cloud Migration Optimization Approaches." In: *Recent Advances on Soft Computing and Data Mining: Proceedings of the First International Conference on Soft Computing and Data Mining (SCDM-2014)*. (Johor, Malaysia, June 16–18, 2014). Ed. by Herawan, T., Ghazali, R., and Deris, M. M. Vol. 287. Advances in Intelligent Systems and Computing. Cham, Switzerland: Springer International Publishing, 2014, pp. 633–645. ISBN: 978-3-319-07691-1. DOI: 10.1007/978-3-319-07692-8_60 (cit. on pp. 21, 24).

[44] Cretella, G. and Martino, B. D. "An Overview of Approaches for the Migration of Applications to the Cloud." In: *Smart Organizations and Smart Artifacts: Fostering Interaction Between People, Technologies and Processes*. Ed. by Caporarello, L., Di Martino, B., and Martinez, M. Lecture Notes in Information Systems and Organisation. Cham, Switzerland: Springer International Publishing, 2014, pp. 67–75. ISBN: 978-3-319-07039-1. DOI: 10.1007/978-3-319-07040-7_8 (cit. on p. 21).

[45] Cardoso, A., Moreira, F., and Simões, P. "A Survey of Cloud Computing Migration Issues and Frameworks." In: *New Perspectives in Information Systems and Technologies, Volume 1*. Ed. by Rocha, Á. et al. Vol. 275. Advances in Intelligent Systems and Computing. Cham, Switzerland: Springer International Publishing, 2014, pp. 161–170. ISBN: 978-3-319-05950-1. DOI: 10.1007/978-3-319-05951-8_16 (cit. on p. 21).

[46] Darsow, A. *The CloudDSF : The Cloud Decision Support Framework*. University of Stuttgart. 2014. URL: http://www.clouddsf.com/ (visited on 09/11/2014) (cit. on pp. 30, 94, 95, 99, 103).

[47] Fowler, M. *Patterns of Enterprise Application Architecture*. The Addison-Wesley Signature Series. Boston, MA, USA: Addison-Wesley, 2003. ISBN: 0-321-12742-0 (cit. on pp. 33, 34).

[48] Microsoft Inc. *Microsoft Application Architecture Guide*. 2nd ed. Patterns & Practices. Redmond, WA, USA: Microsoft, 2009. ISBN: 978-0-7356-2710-9. URL: http://msdn.microsoft.com/en-us/library/ff650706.aspx (visited on 10/04/2014) (cit. on pp. 33, 34).

[49] Oracle Inc. *The Java EE 7 Tutorial: Distributed Multitiered Applications*. 2014. URL: http://docs.oracle.com/javaee/7/tutorial/doc/overview003.htm (visited on 10/04/2014) (cit. on p. 33).

[50] Adler, B. *Building Scalable Applications In the Cloud: Reference Architecture & Best Practices*. RightScale Inc. 2011. URL: http://assets.rightscale.com/uploads/pdfs/Building-Scalable-Applications-in-the-Cloud-White-Paper-by-RightScale.pdf (visited on 10/10/2014) (cit. on pp. 33, 37).

[51] Alonso, G. et al. *Web Services: Concepts, Architectures and Applications*. Data-Centric Systems and Applications. Berlin, Germany: Springer, 2004. ISBN: 3-662-10876-3 (cit. on pp. 35, 70).

[52] Agrawal, D. et al. "Database Scalability, Elasticity, and Autonomy in the Cloud." In: *Database Systems for Advanced Applications: 16th International Conference (DASFAA 2011), Proceedings, Part I*. (Hong Kong, China, Apr. 22–25, 2011). Ed. by Yu, J. X., Kim, M. H., and Unland, R. Vol. 6587. Lecture Notes in Computer Science. Heidelberg, Germany: Springer, 2011, pp. 2–15. ISBN: 978-3-642-20148-6. DOI: 10.1007/978-3-642-20149-3_2 (cit. on p. 37).

[53] Vaquero, L. M., Rodero-Merino, L., and Buyya, R. "Dynamically Scaling Applications in the Cloud." In: *ACM SIGCOMM Computer Communication Review* 41 (1 2011), pp. 45–52. ISSN: 0146-4833. DOI: 10.1145/1925861.1925869 (cit. on pp. 37, 38, 60).

[54] Schwartz, B., Zaitsev, P., and Tkachenko, V. *High performance MySQL*. 3rd ed. Sebastopol, CA, USA: O'Reilly, 2012. ISBN: 978-1-4493-1428-6 (cit. on p. 37).

[55] Strauch, S. et al. "ESB^{MT}: A Multi-tenant Aware Enterprise Service Bus." In: *International Journal of Next-Generation Computing* 4 (3 2013). Perpetual Innovation Media Pvt. Ltd, pp. 230–249. ISSN: 0976-5034. URL: http://ijngc.perpetualinnovation.net (cit. on p. 37).

[56] Cecchet, E., Candea, G., and Ailamaki, A. "Middleware-based Database Replication: The Gaps Between Theory and Practice." In: *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. (Vancouver, Canada, June 9–12, 2008). Ed. by Lakshmanan, L. V. S. New York, NY, USA: ACM, 2008, pp. 739–752. ISBN: 978-1-60558-102-6. DOI: 10.1145/1376616.1376691 (cit. on p. 37).

[57] Anagnostopoulos, V. et al. "Intelligent Clouds: A Middleware Architecture Supporting Business Elasticity." In: *Proceedings of the 18th Panhellenic Conference on Informatics*. (Athens, Greece, Oct. 2–4, 2014). Ed. by Sokratis, K. et al. New York, NY, USA: ACM, 2014, pp. 1–6. ISBN: 978-1-4503-2897-5. DOI: 10.1145/2645791.2645844 (cit. on p. 37).

[58] Caron, E. et al. *Auto-Scaling, Load Balancing and Monitoring in Commercial and Open-Source Clouds*. Research Report RR-7857. 2012. URL: https://hal.inria.fr/hal-00668713 (visited on 02/03/2015) (cit. on pp. 39, 80).

[59] Pors, M. et al. *Sharing is Caring - A Decision Support Model for Multi-Tenant Architectures*. Tech. rep. UU-CS-2013-015. 2013. URL: http://www.cs.uu.nl/research/techreps/repo/CS-2013/2013-015.pdf (visited on 09/08/2014) (cit. on pp. 40, 41).

[60] Guo, C. J. et al. "A Framework for Native Multi-Tenancy Application Development and Management." In: *The 9th IEEE International Conference on E-Commerce Technology and The 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services (CEC-EEE 2007)*. (Tokyo, Japan, July 23–26, 2007). Los Alamitos, CA, USA: IEEE, 2007, pp. 551–558. ISBN: 0-7695-2913-5. DOI: 10.1109/CEC-EEE.2007.4 (cit. on pp. 40–42).

[61] Osipov, C. et al. *Develop and Deploy Multi-Tenant Web-delivered Solutions using IBM middleware: Part 2: Approaches for enabling multi-tenancy*. IBM. 2009. URL: http://www.ibm.com/developerworks/webservices/library/ws-multitenantpart2/index.html (visited on 12/16/2014) (cit. on pp. 41, 42).

[62] Natis, V. Y. *Gartner Reference Model for Elasticity and Multitenancy*. G00231615. 2012. URL: https://www.gartner.com/doc/2058722/gartner-reference-model-elasticity-multitenancy (visited on 12/13/2014) (cit. on pp. 41–44).

[63] Walraven, S., Truyen, E., and Joosen, W. "A Middleware Layer for Flexible and Cost-Efficient Multi-tenant Applications." In: *Middleware 2011: ACM/IFIP/USENIX 12th International Middleware Conference Proceedings*. (Lisbon, Portugal, Dec. 12–16, 2011). Ed. by Hutchison, D. et al. Vol. 7049. Lecture Notes in Computer Science. Berlin, Germany: Springer, 2011, pp. 370–389. ISBN: 978-3-642-25820-6. DOI: 10.1007/978-3-642-25821-3_19 (cit. on p. 41).

[64] Betts, D. et al. *Developing multi-tenant applications for the cloud on Windows Azure*. 3. ed. Patterns & Practices. Redmond, WA, USA: Microsoft, 2012. ISBN: 978-1-62114-023-8 (cit. on pp. 41–43).

[65] Krebs, R., Momm, C., and Kounev, S. "Architectural Concerns in Multi-tenant SaaS Applications." In: *Procceeddings of the 2nd International Conference on Cloud Computing and Services Science (CLOSER 2012)*. (Porto, Potugal, Apr. 18–21, 2012). SciTePress, 2012, pp. 426–431 (cit. on p. 41).

[66] Force.com. *The Force.com Multitenant Architecture: Understanding the Design of Salesforce.com's Internet Application Development Platform*. salesforce.com. 2008. URL: http://www.eurolanresearch.com/wp-content/uploads/2014/05/Force.com_Multitenancy_WP_101508.pdf (visited on 12/13/2014) (cit. on pp. 41, 43, 44).

[67] Chong, F., Carraro, G., and Wolter, R. *Multi-Tenant Data Architecture*. Microsoft Inc. 2006. URL: http://msdn.microsoft.com/en-us/library/aa479086.aspx (visited on 12/16/2014) (cit. on p. 41).

[68] Lazri, K., Laniepce, S., and Ben-Othman, J. "When Dynamic VM Migration Falls under the Control of VM Users." In: *IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom 2013)*. (Bristol, UK, Dec. 2–5, 2013). Vol. 1. Piscataway, NJ, USA: IEEE Computer Society, 2013, pp. 395–402. ISBN: 978-0-7695-5095-4. DOI: 10.1109/CloudCom.2013.58 (cit. on pp. 42, 58).

[69] AlJahdali, H. et al. "Multi-tenancy in Cloud Computing." In: *2014 IEEE 8th International Symposium on Service Oriented System Engineering (SOSE)*. (Oxford, UK, Apr. 7–11, 2014). IEEE Computer Society, 2014, pp. 344–351. ISBN: 978-1-4799-2504-9. DOI: 10.1109/SOSE.2014.50 (cit. on pp. 42, 58).

[70] Kabbedijk, J. et al. "Multi-tenant Architecture Comparison." In: *Software architecture: 8th European Conference Procceedings, ECSA 2014*. (Vienna, Austria, Aug. 25–29, 2014). Ed. by Avgeriou, P. and Zdun, U. Vol. 8627. Lecture Notes in Computer Science. Springer International Publishing, 2014, pp. 202–209. ISBN: 978-3-319-09969-9. DOI: 10.1007/978-3-319-09970-5_18. URL: http://dx.doi.org/10.1007/978-3-319-09970-5_18 (cit. on p. 43).

[71] Wainewright, P. *Microsoft nudges ERP into the cloud*. 2013. URL: http://diginomica.com/2013/11/13/microsoft-nudges-erp-cloud/ (visited on 12/16/2014) (cit. on p. 44).

[72] Momm, C. and Krebs, R. "A Qualitative Discussion of Different Approaches for Implementing Multi-Tenant SaaS Offerings." In: *Software Engineering 2011 – Workshopband (ESoSyM-2011)*. (Karlsruhe, Germany, Feb. 21, 2011). Ed. by Pretschner, A., Reussner, R., and Jähnichen, S. Vol. 184. Fachgruppe OOSE der Gesellschaft für Informatik und ihrer Arbeitskreise. Bonn, Germany: Bonner Köllen Verlag, 2011. ISBN: 978-3-88579-278-9 (cit. on p. 44).

[73] Cloud Special Interest Group and Payment Card Industry Security Standards Council. *PCI DSS Cloud Computing Guidelines*. 2013. URL: https://www.pcisecuritystandards.org/pdfs/PCI_DSS_v2_Cloud_Guidelines.pdf (visited on 12/19/2014) (cit. on pp. 45, 53).

[74] Microsoft Inc. *Unveiling The Microsoft Cloud Platform System, powered by Dell*. Microsoft Inc. 2014. URL: http://blogs.technet.com/b/windowsserver/archive/2014/10/20/unveiling-the-microsoft-cloud-platform-system-powered-by-dell.aspx (visited on 12/23/2014) (cit. on p. 46).

[75] Oracle Inc. *Oracle Delivers Oracle Infrastructure as a Service on Premise with Capacity on Demand*. Oracle Inc. 2013. URL: http://www.oracle.com/us/corporate/press/1897474 (visited on 12/17/2014) (cit. on pp. 46, 53).

[76] Amazon Inc. *Amazon Web Services: How AWS Pricing Works July 2014*. Amazon Inc. 2014. URL: https://media.amazonwebservices.com/AWS_Pricing_Overview.pdf (visited on 12/25/2014) (cit. on pp. 48, 53).

[77] Technology Research Project Corporate. *Report on Cloud Data Regulations: A contribution on how to reduce the compliancy costs of Cross-Border Data Transfers*. 2014. URL: http://asiacloudcomputing.org/research/2014-cloud-data-regulations (visited on 12/19/2014) (cit. on p. 48).

[78] Castro, D. *The False Promise of Data Nationalism*. Information Technology and Innovation Foundation. 2013. URL: http://www2.itif.org/2013-false-promise-data-nationalism.pdf (visited on 12/25/2014) (cit. on p. 48).

[79] Dimension Data. *Pursuing Compliance in Public Cloud: Identifying the right compliance strategy for your business in the cloud*. 2013. URL: http://www.dimensiondata.com/en-LU/Pages/Profile%20Boxes/Dimension-Data-Pursuing-Compliance-in-Public-Cloud.aspx (visited on 12/25/2014) (cit. on p. 48).

[80] Hon, W. K. and Millard, C. "Data Export in Cloud Computing – How Can Personal Data Be Transferred Outside the EEA? The Cloud of Unknowing, Part 4." In: *SSRN Electronic Journal* 9:1 (25 2012). Queen Mary School of Law Legal Studies Research Paper No. 85/2011. ISSN: 1556-5068. DOI: 10.2139/ssrn.1925066 (cit. on p. 48).

[81] Talbot, C. *Amazon Introduces Annual Subscription Pricing to Marketplace*. 2014. URL: http://talkincloud.com/iaas/071614/amazon-introduces-annual-subscription-pricing-marketplace (visited on 12/17/2014) (cit. on p. 53).

[82] Hosseini, H. *AWS vs Google Pricing: Decoding the New AWS RI Model*. RightScale Inc. 2014. URL: http://www.rightscale.com/blog/cloud-cost-analysis/aws-vs-google-pricing-decoding-new-aws-ri-model (visited on 12/25/2014) (cit. on p. 53).

[83] Liu, F. et al. *NIST Cloud Computing Reference Architecture*. Vol. 500-292. NIST Special Publication. Gaithersburg, MD, USA: NIST, 2011. ISBN: 978-1-4781-6802-7. URL: http://www.nist.gov/manuscript-publication-search.cfm?pub_id=909505 (cit. on p. 53).

[84] Melnik, G. *Windows Azure autoscaling now built-in*. 2013. URL: http://blogs.msdn.com/b/agile/archive/2013/07/02/windows-azure-autoscaling-now-built-in.aspx (visited on 12/26/2014) (cit. on p. 57).

[85] Synytsky, R. *The Truth About PaaS Vertical Scaling and Why You are Being Oversold*. Jelastic Inc. 2013. URL: http://java.dzone.com/articles/truth-about-paas-vertical (visited on 01/13/2015) (cit. on p. 78).

[86] Brebner, P. C. "Is your cloud elastic enough?: performance modelling the elasticity of infrastructure as a service (IaaS) cloud applications." In: *Proceedings of the 3rd Joint WOSP/SIPEW International Conference on Performance Engineering*. (Boston, MA, USA, Apr. 22–25, 2012). Ed. by Kaeli, D. and Rolia, J. 2012, pp. 263–266. ISBN: 978-1-4503-1202-8. DOI: 10.1145/2188286.2188334 (cit. on p. 80).

[87] Galante, G. and Bona, L. C. d. "A Survey on Cloud Computing Elasticity." In: *Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing*. (Chicago, IL, USA, Nov. 5–8, 2012). Los Alamitos, CA, USA: IEEE Computer Society, 2012, pp. 263–270. ISBN: 978-1-4673-4432-6. DOI: 10.1109/UCC.2012.30 (cit. on p. 80).

[88] Lorido-Botrán, T., Miguel-Alonso, J., and Lozano, J. A. "A Review of Auto-scaling Techniques for Elastic Applications in Cloud Environments." In: *Journal of Grid Computing* 12 (4 2014), pp. 559–592. ISSN: 1570-7873. DOI: 10.1007/s10723-014-9314-7 (cit. on pp. 80, 81).

[89] Ahn, Y. et al. "An auto-scaling mechanism for virtual resources to support mobile, pervasive, real-time healthcare applications in cloud computing." In: *IEEE Network* 27 (5 2013), pp. 62–68. ISSN: 0890-8044. DOI: 10.1109/MNET.2013.6616117 (cit. on p. 80).

[90] Jamshidi, P., Ahmad, A., and Pahl, C. "Autonomic resource provisioning for cloud-based software." In: *9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. (Hyderabad, India, June 2–3, 2014). Ed. by Engels, G. and Bencomo, N. 2014, pp. 95–104. ISBN: 978-1-4503-2864-7. DOI: 10.1145/2593929.2593940 (cit. on pp. 80, 81).

[91] Lu, L. et al. “Application-driven dynamic vertical scaling of virtual machines in resource pools.” In: *IEEE/IFIP Network Operations and Management Symposium*. (Krakow, Poland, May 5–9, 2014). Piscataway, NJ, USA: IEEE, 2014, pp. 1–9. ISBN: 978-1-4799-0913-1. DOI: 10.1109/NOMS.2014.6838238 (cit. on p. 80).

[92] Sallam, A. and Li, K. “Virtual Machine Proactive Scaling in Cloud Systems.” In: *2012 IEEE International Conference on Cluster Computing Workshops*. (Beijing, China, Sept. 24–28, 2012). Los Alamitos, CA, USA: Conference Publishing Services, 2012, pp. 97–105. ISBN: 978-1-4673-2893-7. DOI: 10.1109/ClusterW.2012.17 (cit. on p. 80).

[93] Bunch, C. et al. “A Pluggable Autoscaling Service for Open Cloud PaaS Systems.” In: *The 5th IEEE/ACM International Conference on Utility and Cloud Computing (UCC 2012)*. (Chicago, IL, USA, Nov. 5–8, 2012). Los Alamitos, CA, USA: IEEE Computer Society, 2012, pp. 191–194. ISBN: 978-0-7695-4862-3. DOI: 10.1109/UCC.2012.12 (cit. on p. 80).

[94] Verma, M. et al. “Resource Demand Prediction in Multi-Tenant Service Clouds.” In: *2013 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*. (Bangalore, India, Oct. 16–18, 2013). 2013, pp. 1–8. ISBN: 978-1-4799-0027-5. DOI: 10.1109/CCEM.2013.6684440 (cit. on p. 80).

[95] Ardagna, C. A. et al. “Scalability Patterns for Platform-as-a-Service.” In: *2012 IEEE 5th International Conference on Cloud Computing*. (Honolulu, HI, USA, June 24–29, 2012). Ed. by Chang, R. Los Alamitos, CA, USA: IEEE Computer Society, 2012, pp. 718–725. ISBN: 978-0-7695-4755-8. DOI: 10.1109/CLOUD.2012.41 (cit. on p. 80).

[96] RightScale Inc. *Cloud Pricing Trends*. RightScale Inc. 2014. URL: http://www.rightscale.com/blog/cloud-industry-insights/cloud-price-reductions-definitive-analysis-2013-trends (visited on 02/01/2015) (cit. on pp. 114, 115).

[97] Hosseini, H. *AWS Responds with Price Cuts: Google vs AWS Pricing Round 2*. RightScale Inc. 2014. URL: http://www.rightscale.com/blog/cloud-cost-analysis/aws-responds-price-cuts-google-vs-aws-pricing-round-2 (visited on 02/02/2015) (cit. on p. 114).

[98] Leong, L. et al. *Magic Quadrant for Cloud Infrastructure as a Service*. ID: G00261698. Gartner Inc. 2014. URL: http://www.gartner.com/technology/reprints.do?id=1-1UKQQA6%5C&ct=140528%5C&st=sb (visited on 02/01/2015) (cit. on p. 115).

A. Appendix

A.1. CloudDSF+ Knowledge Base

In the following all tables showing the relations between decisions and between outcomes are listed. In order to keep the tables concise and informative, wherever possible, not existing relations and thus empty fields have been omitted.

Table A.1.: Influencing (I), affecting (A) and binding (B) relations between decisions.

Decisions	Select Application Layer	Select Application Tier	Select Application Components	Select Migration Type	Define Scalability Level	Select Scaling Type	Select Elasticity Automation Degree	Select Scaling Trigger	Select Multi-Tenancy Level	Select Cloud Deployment Model	Select Cloud Service Model	Define Cloud Hosting	Define Roles of Responsibility	Select Cloud Vendor	Select Pricing Model	Define Resource Location
Select Application Layer		–	I	I	–	–	–	–	I	–	–	–	–	–	–	–
Select Application Tier	–		–	–	–	–	–	–	–	–	–	–	–	–	–	–
Select Application Components	I	–		I	I	–	–	–	I	–	I	–	–	–	–	–
Select Migration Type	I	–	I		I	–	–	–	I	–	I	–	–	–	–	–
Define Scalability Level	–	–	I	I		I	I	I	I	–	I	–	–	A	–	–
Select Scaling Type	–	–	–	–	I		–	–	–	–	I	–	–	A	–	–
Select Elasticity Automation Degree	–	–	–	–	I	–		I	–	–	I	–	–	A	–	–
Select Scaling Trigger	–	–	–	–	I	–	I		–	–	I	–	–	A	–	–
Select Multi-Tenancy Level	I	–	I	I	I	–	–	–		–	I	–	–	A	–	–
Select Cloud Deployment Model	–	–	–	–	–	–	–	–	–		–	I	I	A	–	–
Select Cloud Service Model	–	–	I	I	I	I	I	I	I	–		–	–	A	–	–
Define Cloud Hosting	–	–	–	–	–	–	–	–	–	I	–		I	A	–	I
Define Roles of Responsibility	–	–	–	–	–	–	–	–	–	I	–	I		A	–	–
Select Cloud Vendor	–	–	–	–	B	B	B	B	B	B	B	B	B		B	B
Select Pricing Model	–	–	–	–	–	–	–	–	–	–	–	–	–	A		–
Define Resource Location	–	–	–	–	–	–	–	–	–	–	–	I	–	A	–	

Table A.2.: Requiring (R) relations between decisions.

Decisions	Select Application Layer	Select Application Tier	Select Application Components	Select Migration Type	Define Scalability Level	Select Scaling Type	Select Elasticity Automation Degree	Select Scaling Trigger	Select Multi-Tenancy Level	Select Cloud Deployment Model	Select Cloud Service Model	Define Cloud Hosting	Define Roles of Responsibility	Select Cloud Vendor	Select Pricing Model	Define Resource Location
Select Application Layer		–	R	–	–	–	–	–	–	–	–	–	–	–	–	–
Select Application Tier	–		R	–	–	–	–	–	–	–	–	–	–	–	–	–
Select Application Components	–	–		R	–	–	–	–	–	–	R	–	–	–	–	–
Select Migration Type	–	–	R		–	–	–	–	–	–	–	–	–	–	–	–
Define Scalability Level	–	–	–	–		–	–	–	–	–	R	–	–	–	–	–
Select Scaling Type	–	–	–	–	R		–	–	–	–	–	–	–	–	–	–
Select Elasticity Automation Degree	–	–	–	–	–	R		R	–	–	–	–	–	–	–	–
Select Scaling Trigger	–	–	–	–	–	–	R		–	–	–	–	–	–	–	
Select Multi-Tenancy Level	–	–	–	–	–	–	–	–		–	R	–	–	–	–	–
Select Cloud Deployment Model	–	–	–	–	–	–	–	–	–		–	R	–	R	–	–
Select Cloud Service Model	–	–	–	–	–	–	–	–	–	–		–	–	–	–	–
Define Cloud Hosting	–	–	–	–	–	–	–	–	–	–	–		–	–	–	R
Define Roles of Responsibility	–	–	–	–	–	–	–	–	–	–	–	–		–	–	–
Select Cloud Vendor	–	–	–	–	–	–	–	–	–	–	–	–	–		–	–
Select Pricing Model	–	–	–	–	–	–	–	–	–	–	–	–	–	–		–
Define Resource Location	–	–	–	–	–	–	–	–	–	–	–	R	–	–	–	

Table A.3.: Outcome Relations of *Select Application Layer*

Select Application Layer	Application Component	Application Components	Middleware Component	Middleware Components	Application + Middleware Component	Application Component + Middleware Components	Middleware Component + Application Components	Application + Middleware Components	Migration Type I	Migration Type II	Migration Type III	Migration Type IV	Shared Hardware Multi-Tenancy	Shared OS Multi-Tenancy	Shared Middleware Multi-Tenancy	Shared Application Multi-Tenancy
	Select Application Components								**Select Migration Type**				**Select Multi-Tenancy Level**			
Presentation Layer	a	a	ex	ex	a	a	a	a	a	a	ex	ex	ex	ex	ex	ex
Business Layer	a	a	a	a	a	a	a	a	a	a	ex	ex	ex	ex	ex	ex
Resource Layer	ex	ex	a	a	ex	ex	ex	ex	a	a	ex	ex	ex	ex	ex	ex
Presentation + Business Layer	ex	a	ex	ex	a	a	a	a	ex	a	ex	ex	ex	ex	ex	ex
Presentation + Resource Layer	ex	ex	ex	ex	a	a	a	a	ex	a	ex	ex	ex	ex	ex	ex
Business + Resource Layer	ex	ex	ex	a	a	a	a	a	ex	a	ex	ex	ex	ex	ex	ex
Presentation + Business + Resource Layer	ex	ex	ex	ex	ex	a	a	a	ex	a	a	a	a	a	a	a

Table A.4.: Outcome Relations of *Select Application Components* 1-2

Select Application Components	Presentation Layer	Business Layer	Resource Layer	Presentation + Business Layer	Presentation + Resource Layer	Business + Resource Layer	Presentation + Business + Resource Layer	Migration Type I	Migration Type II	Migration Type III	Migration Type IV	Shared Hardware Multi-Tenancy	Shared OS Multi-Tenancy	Shared Middleware Multi-Tenancy	Shared Application Multi-Tenancy
	Select Application Layer							**Select Migration Type**				**Select Multi-Tenancy Level**			
Application Component	a	a	ex	ex	ex	ex	ex	a	ex	ex	ex	ex	ex	ex	ex
Application Components	a	a	ex	a	ex	ex	ex	ex	a	ex	ex	ex	ex	ex	ex
Middleware Component	ex	a	a	ex	ex	ex	ex	a	ex	ex	ex	ex	ex	ex	ex
Middleware Components	ex	a	a	ex	ex	a	ex	ex	a	ex	ex	ex	ex	ex	ex
Application + Middleware Component	a	a	ex	a	a	a	ex	ex	a	ex	ex	ex	ex	ex	ex
Application Component + Middleware Components	a	a	ex	a	a	a	a	ex	a	ex	ex	ex	ex	ex	ex
Middleware Component + Application Components	a	a	ex	a	a	a	a	ex	a	ex	ex	ex	ex	ex	ex
Application + Middleware Components	a	a	ex	a	a	a	a	ex	a	a	a	a	a	a	a

Table A.5.: Outcome Relations of *Select Application Components* 2-2

Select Application Components	Define Scalability Level								Select Cloud Service Model						
	No Scaling	VM Level Scaling	Middleware Level Scaling	Application Level Scaling	VM + Middleware Level Scaling	VM + Application Level Scaling	Middleware + Application Level Scaling	VM + Middleware + Application Level Scaling	IaaS	PaaS	SaaS	IaaS + PaaS	IaaS + SaaS	PaaS + SaaS	Iaas + PaaS + SaaS
Application Component	a	ex	ex	a	ex	ex	ex	ex	a	a	a	ex	ex	ex	ex
Application Components	a	ex	ex	a	ex	ex	ex	ex	a	a	a	a	a	a	a
Middleware Component	a	a	a	ex	a	ex	ex	ex	a	a	ex	ex	ex	ex	ex
Middleware Components	a	a	a	ex	a	ex	ex	ex	a	a	ex	a	ex	ex	ex
Application + Middleware Component	a	a	a	a	a	a	a	a	a	a	ex	a	ex	ex	ex
Application Component + Middleware Components	a	a	a	a	a	a	a	a	a	a	ex	a	ex	ex	ex
Middleware Component + Application Components	a	a	a	a	a	a	a	a	a	a	ex	a	ex	ex	ex
Application + Middleware Components	a	a	a	a	a	a	a	a	a	a	ex	a	ex	ex	ex

Table A.6.: Outcome Relations of *Select Migration Type* 1-2

Select Migration Type	Select Application Layer							Select Application Components							
	Presentation Layer	Business Layer	Resource Layer	Presentation + Business Layer	Presentation + Resource Layer	Business + Resource Layer	Presentation + Business + Resource Layer	Application Component	Application Components	Middleware Component	Middleware Components	Application + Middleware Component	Application Component + Middleware Components	Middleware Component + Application Components	Application + Middleware Components
Migration Type I	a	a	a	ex	ex	ex	ex	a	ex	a	ex	ex	ex	ex	ex
Migration Type II	a	a	a	a	a	a	a	ex	a	ex	a	a	a	a	a
Migration Type III	ex	ex	ex	ex	ex	ex	in	ex	ex	ex	ex	ex	ex	ex	in
Migration Type IV	ex	ex	ex	ex	ex	ex	in	ex	ex	ex	ex	ex	ex	ex	in

Table A.7.: Outcome Relations of *Select Migration Type* 2-2

Select Migration Type	No Scaling	VM Level Scaling	Middleware Level Scaling	Application Level Scaling	VM + Middleware Level Scaling	VM + Application Level Scaling	Middleware + Application Level Scaling	VM + Middleware + Application Level Scaling	Shared Hardware Multi-Tenancy	Shared OS Multi-Tenancy	Shared Middleware Multi-Tenancy	Shared Application Multi-Tenancy	IaaS	PaaS	SaaS	IaaS + PaaS	IaaS + SaaS	PaaS + SaaS	Iaas + PaaS + SaaS
	Define Scalability Level								**Select Multi-Tenancy Level**				**Select Cloud Service Model**						
Migration Type I	a	a	a	a	a	ex	ex	ex	ex	ex	ex	ex	a	a	a	ex	ex	ex	ex
Migration Type II	a	a	a	a	a	a	a	a	ex	ex	ex	ex	a	a	a	a	a	a	a
Migration Type III	a	a	a	a	a	a	a	a	a	a	a	a	in	ex	ex	ex	ex	ex	ex
Migration Type IV	ex	a	a	a	a	a	a	a	a	a	a	a	ex	a	a	ex	ex	a	ex

Table A.8.: Outcome Relations of *Define Scalability Level* 1-3

Define Scalability Level	Application Component	Application Components	Middleware Component	Middleware Components	Application + Middleware Component	Application Component + Middleware Components	Middleware Component + Application Components	Application + Middleware Components	Migration Type I	Migration Type II	Migration Type III	Migration Type IV	Vertical Scaling	Horizontal Scaling	Hybrid Scaling
	Select Application Components								**Select Migration Type**				**Select Scaling Type**		
No Scaling	a	a	a	a	a	a	a	a	a	a	a	ex	ex	ex	ex
VM Level Scaling	ex	ex	a	a	a	a	a	a	a	a	a	a	a	a	a
Middleware Level Scaling	ex	ex	a	a	a	a	a	a	a	a	a	a	ex	a	ex
Application Level Scaling	a	a	ex	ex	a	a	a	a	a	a	a	a	ex	a	ex
VM + Middleware Level Scaling	ex	ex	a	a	a	a	a	a	a	a	a	a	ex	a	a
VM + Application Level Scaling	ex	ex	ex	ex	a	a	a	a	ex	a	a	a	ex	a	a
Middleware + Application Level Scaling	ex	ex	ex	ex	a	a	a	a	ex	a	a	a	ex	a	ex
VM + Middleware + Application Level Scaling	ex	ex	ex	ex	a	a	a	a	ex	a	a	a	ex	a	a

Table A.9.: Outcome Relations of *Define Scalability Level* 2-3

Define Scalability Level	Manual Scaling	Semi-Automatic Scaling	Semi-Automatic Third-Party Scaling	Automatic Scaling	Automatic Third-Party Scaling	No Trigger	Event-Driven Trigger	Proactive Trigger	Shared Hardware Multi-Tenancy	Shared OS Multi-Tenancy	Shared Middleware Multi-Tenancy	Shared Application Multi-Tenancy
	Select Elasticity Automation Degree					Select Scaling Trigger			Select Multi-Tenancy Level			
No Scaling	ex	ex	ex	ex	ex	ex	ex	ex	ex	ex	ex	ex
VM Level Scaling	a	a	a	a	a	a	a	a	a	a	ex	ex
Middleware Level Scaling	a	a	a	ex	a	a	a	a	ex	ex	ex	ex
Application Level Scaling	a	a	a	ex	a	a	a	a	ex	ex	ex	ex
VM + Middleware Level Scaling	a	a	a	ex	a	a	a	a	a	a	a	ex
VM + Application Level Scaling	a	a	a	ex	a	a	a	a	a	a	ex	ex
Middleware + Application Level Scaling	a	a	a	ex	a	a	a	a	ex	ex	ex	ex
VM + Middleware + Application Level Scaling	a	a	a	ex	a	a	a	a	a	a	a	a

Table A.10.: Outcome Relations of *Define Scalability Level* 3-3

Define Scalability Level	IaaS	PaaS	SaaS	IaaS + PaaS	IaaS + SaaS	PaaS + SaaS	Iaas + PaaS + SaaS	Evaluated Cloud Vendor
	Select Cloud Service Model							Select Cloud Vendor
No Scaling	a	a	ex	a	a	a	a	aff
VM Level Scaling	a	ex	ex	a	a	ex	a	aff
Middleware Level Scaling	a	ex	ex	a	a	ex	a	aff
Application Level Scaling	a	a	ex	a	a	a	a	aff
VM + Middleware Level Scaling	a	ex	ex	a	a	ex	a	aff
VM + Application Level Scaling	a	ex	ex	a	a	ex	a	aff
Middleware + Application Level Scaling	a	ex	ex	a	a	ex	a	aff
VM + Middleware + Application Level Scaling	a	ex	ex	a	a	ex	a	aff

Table A.11.: Outcome Relations of *Select Elasticity Automation Degree*

Select Elasticity Automation Degree	Define Scalability Level								Select Scaling Trigger			Select Cloud Service Model							Select Cloud Vendor
	No Scaling	VM Level Scaling	Middleware Level Scaling	Application Level Scaling	VM + Middleware Level Scaling	VM + Application Level Scaling	Middleware + Application Level Scaling	VM + Middleware + Application Level Scaling	No Trigger	Event-Driven Trigger	Proactive Trigger	IaaS	PaaS	SaaS	IaaS + PaaS	IaaS + SaaS	PaaS + SaaS	Iaas + PaaS + SaaS	Evaluated Cloud Vendor
Manual Scaling	ex	a	a	a	a	a	a	a	in	ex	ex	a	a	ex	a	a	a	a	aff
Semi-Automatic Scaling	ex	a	a	a	a	a	a	a	ex	in	ex	a	a	ex	a	a	a	a	aff
Semi-Automatic Third-Party Scaling	ex	a	a	a	a	a	a	a	ex	ex	ex	a	a	ex	a	a	a	a	aff
Automatic Scaling	ex	a	ex	ex	ex	ex	ex	ex	ex	a	a	a	ex	ex	a	a	ex	a	aff
Automatic Third-Party Scaling	ex	a	a	a	a	a	a	a	ex	ex	ex	a	a	ex	a	a	a	a	aff

Table A.12.: Outcome Relations of *Select Scaling Trigger*

Select Scaling Trigger	Define Scalability Level								Select Elasticity Automation Degree					Select Cloud Service Model							Select Cloud Vendor
	No Scaling	VM Level Scaling	Middleware Level Scaling	Application Level Scaling	VM + Middleware Level Scaling	VM + Application Level Scaling	Middleware + Application Level Scaling	VM + Middleware + Application Level Scaling	Manual Scaling	Semi-Automatic Scaling	Semi-Automatic Third-Party Scaling	Automatic Scaling	Automatic Third-Party Scaling	IaaS	PaaS	SaaS	IaaS + PaaS	IaaS + SaaS	PaaS + SaaS	Iaas + PaaS + SaaS	Evaluated Cloud Vendor
No Trigger	ex	a	a	a	a	a	a	a	in	ex	ex	ex	ex	a	a	ex	a	a	a	a	aff
Event-Driven Trigger	ex	a	a	a	a	a	a	a	ex	a	ex	a	ex	a	a	ex	a	a	a	a	aff
Proactive Trigger	ex	a	a	a	a	a	a	a	ex	ex	ex	a	ex	a	ex	ex	a	a	ex	a	aff

Table A.13.: Outcome Relations of *Select Scaling Type*

Select Scaling Type	No Scaling	VM Level Scaling	Middleware Level Scaling	Application Level Scaling	VM + Middleware Level Scaling	VM + Application Level Scaling	Middleware + Application Level Scaling	VM + Middleware + Application Level Scaling	IaaS	PaaS	SaaS	IaaS + PaaS	IaaS + SaaS	PaaS + SaaS	Iaas + PaaS + SaaS	Evaluated Cloud Vendor
	Define Scalability Level								**Select Cloud Service Model**							**Select Cloud Vendor**
Vertical Scaling	ex	in	ex	ex	ex	ex	ex	ex	a	ex	ex	a	a	ex	a	aff
Horizontal Scaling	ex	a	a	a	a	a	a	a	a	a	ex	a	a	a	a	aff
Hybrid Scaling	ex	a	ex	ex	a	a	ex	a	a	ex	ex	a	a	ex	a	aff

Table A.14.: Outcome Relations of *Select Multi-Tenancy Level* 1-2

Select Multi-Tenancy Level	Presentation Layer	Business Layer	Resource Layer	Presentation + Business Layer	Presentation + Resource Layer	Business + Resource Layer	Presentation + Business + Resource Layer	Application Component	Application Components	Middleware Component	Middleware Components	Application + Middleware Component	Application Component + Middleware Components	Middleware Component + Application Components	Application + Middleware Components	Migration Type I	Migration Type II	Migration Type III	Migration Type IV
	Select Application Layer							**Select Application Components**								**Select Migration Type**			
Shared Hardware Multi-Tenancy	ex	ex	ex	ex	ex	ex	in	ex	ex	ex	ex	ex	ex	ex	in	ex	ex	a	a
Shared OS Multi-Tenancy	ex	ex	ex	ex	ex	ex	in	ex	ex	ex	ex	ex	ex	ex	in	ex	ex	a	a
Shared Middleware Multi-Tenancy	ex	ex	ex	ex	ex	ex	in	ex	ex	ex	ex	ex	ex	ex	in	ex	ex	a	a
Shared Application Multi-Tenancy	ex	ex	ex	ex	ex	ex	in	ex	ex	ex	ex	ex	ex	ex	in	ex	ex	a	a

Table A.15.: Outcome Relations of *Select Multi-Tenancy Level* 2-2

	No Scaling	VM Level Scaling	Middleware Level Scaling	Application Level Scaling	VM + Middleware Level Scaling	VM + Application Level Scaling	Middleware + Application Level Scaling	VM + Middleware + Application Level Scaling	IaaS	PaaS	SaaS	IaaS + PaaS	IaaS + SaaS	PaaS + SaaS	Iaas + PaaS + SaaS	Evaluated Cloud Vendor
Select Multi-Tenancy Level	**Define Scalability Level**								**Select Cloud Service Model**							**Select Cloud Vendor**
Shared Hardware Multi-Tenancy	ex	a	ex	ex	a	a	ex	a	in	ex	ex	ex	ex	ex	ex	aff
Shared OS Multi-Tenancy	ex	a	ex	ex	a	a	ex	a	in	ex	ex	ex	ex	ex	ex	aff
Shared Middleware Multi-Tenancy	ex	ex	ex	ex	a	ex	ex	a	a	a	ex	a	ex	ex	ex	aff
Shared Application Multi-Tenancy	ex	ex	ex	ex	ex	ex	ex	a	a	a	a	a	a	a	a	aff

Table A.16.: Outcome Relations of *Select Cloud Deployment Model*

	On-Premise Hosting	Off-Premise Hosting	Hybrid Hosting	Inhouse	Managmenet	Outsourced	Inhouse + Management	Inhouse + Outsourced	Management + Outsourced	Inhouse + Management + Outsourced	Evaluated Cloud Vendor
Select Cloud Deployment Model	**Define Cloud Hosting**			**Define Roles of Responsibility**							**Select Cloud Vendor**
Public Cloud	ex	in	ex	ex	a	a	ex	ex	ex	ex	aff
Private Cloud	a	a	ex	a	a	a	ex	ex	ex	ex	aff
Community Cloud	ex	ex	a	ex	ex	ex	ex	in	ex	ex	aff
Hybrid Cloud	ex	a	a	ex	a	a	a	a	a	a	aff

Table A.17.: Outcome Relations of *Select Cloud Service Model* 1-2

Select Cloud Service Model	Select Application Components										Select Migration Type					Define Scalability Level								
	Application Component	Application Components	Middleware Component	Middleware Components	Application + Middleware Component	Application Component + Middleware Components	Middleware Component + Application Components	Application + Middleware Components	Migration Type I	Migration Type II	Migration Type III	Migration Type IV	No Scaling	VM Level Scaling	Middleware Level Scaling	Application Level Scaling	VM + Middleware Level Scaling	VM + Application Level Scaling	Middleware + Application Level Scaling	VM + Middleware + Application Level Scaling
IaaS	a	a	a	a	a	a	a	a	a	a	a	ex	a	a	a	a	a	a	a	a
PaaS	a	a	a	a	a	a	a	a	a	a	ex	a	a	ex	ex	a	ex	ex	ex	ex
SaaS	a	a	ex	ex	ex	ex	ex	ex	a	a	ex	a	ex	ex	ex	ex	ex	ex	ex	ex
IaaS + PaaS	ex	a	ex	a	a	a	a	a	ex	a	ex	ex	a	a	a	a	a	a	a	a
IaaS + SaaS	ex	a	ex	ex	ex	ex	ex	ex	ex	a	ex	ex	a	a	a	a	a	a	a	a
PaaS + SaaS	ex	a	ex	ex	ex	ex	ex	ex	ex	a	ex	a	a	ex	ex	a	ex	ex	ex	ex
Iaas + PaaS + SaaS	ex	a	ex	ex	ex	ex	ex	ex	ex	a	ex	ex	a	a	a	a	a	a	a	a

Table A.18.: Outcome Relations of *Select Cloud Service Model* 2-2

Select Cloud Service Model	Select Scaling Type				Select Elasticity Automation Degree					Select Scaling Trigger					Select Multi-Tenancy Level				Select Cloud Vendor
	Vertical Scaling	Horizontal Scaling	Hybrid Scaling	Manual Scaling	Semi-Automatic Scaling	Semi-Automatic Third-Party Scaling	Automatic Scaling	Automatic Third-Party Scaling	No Trigger	Event-Driven Trigger	Proactive Trigger	Shared Hardware Multi-Tenancy	Shared OS Multi-Tenancy	Shared Middleware Multi-Tenancy	Shared Application Multi-Tenancy	Evaluated Cloud Vendor
IaaS	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	aff
PaaS	ex	a	a	a	a	a	ex	a	a	a	ex	ex	ex	a	a	aff
SaaS	ex	ex	ex	ex	ex	ex	ex	ex	ex	ex	ex	ex	ex	ex	a	aff
IaaS + PaaS	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	aff
IaaS + SaaS	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	aff
PaaS + SaaS	ex	a	ex	a	a	a	ex	a	a	a	ex	a	a	a	a	aff
Iaas + PaaS + SaaS	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	aff

Table A.19.: Outcome-Relations of *Define Cloud Hosting*

	Public Cloud	Private Cloud	Community Cloud	Hybrid Cloud	Inhouse	Managmenet	Outsourced	Inhouse + Management	Inhouse + Outsourced	Management + Outsourced	Inhouse + Management + Outsourced	Evaluated Cloud Vendor	Data In Same Jurisdiction	Data In Different Jurisdiction
Define Cloud Hosting	**Select Cloud Deployment Model**				**Define Roles of Responsibility**							**Select Cloud Vendor**	**Define Resource Location**	
On-Premise Hosting	ex	a	ex	ex	in	ex	ex	ex	ex	ex	ex	aff	in	ex
Off-Premise Hosting	a	a	ex	a	ex	a	a	ex	ex	a	ex	aff	a	a
Hybrid Hosting	ex	ex	a	a	ex	ex	ex	a	a	ex	a	aff	a	a

Table A.20.: Outcome-Relations of *Define Roles of Responsibility*

	Public Cloud	Private Cloud	Community Cloud	Hybrid Cloud	On-Premise Hosting	Off-Premise Hosting	Hybrid Hosting	Evaluated Cloud Vendor
Define Roles of Responsibility	**Select Cloud Deployment Model**				**Define Cloud Hosting**			**Select Cloud Vendor**
Inhouse	ex	a	ex	ex	in	ex	ex	aff
Managmenet	a	a	ex	a	ex	in	ex	aff
Outsourced	a	a	ex	a	ex	in	ex	aff
Inhouse + Management	ex	ex	ex	a	ex	ex	in	aff
Inhouse + Outsourced	ex	ex	a	a	ex	ex	in	aff
Management + Outsourced	ex	ex	ex	a	ex	in	ex	aff
Inhouse + Management + Outsourced	ex	ex	ex	a	ex	ex	in	aff

Table A.21.: Outcome Relations of *Define Resource Location*.

	On-Premise Hosting	Off-Premise Hosting	Hybrid Hosting	Evaluated Cloud Vendor
Define Resource Location	**Define Cloud Hosting**			**Select Cloud Vendor**
Data In Same Jurisdiction	a	a	a	aff
Data In Different Jurisdiction	ex	a	a	aff

Table A.22.: Outcome Relations of *Select Cloud Vendor* 1-3

	No Scaling	VM Level Scaling	Middleware Level Scaling	Application Level Scaling	VM + Middleware Level Scaling	VM + Application Level Scaling	Middleware + Application Level Scaling	VM + Middleware + Application Level Scaling	Vertical Scaling	Horizontal Scaling	Hybrid Scaling	Manual Scaling	Semi-Automatic Scaling	Semi-Automatic Third-Party Scaling	Automatic Scaling	Automatic Third-Party Scaling	No Trigger	Event-Driven Trigger	Proactive Trigger
Select Cloud Vendor	**Define Scalability Level**								**Select Scaling Type**			**Select Elasticity Automation Degree**					**Select Scaling Trigger**		
Evaluated Cloud Vendor	eb	eb	eb	eb	eb	eb	eb	eb	eb	eb	eb	eb	eb	eb	eb	eb	eb	eb	eb

Table A.23.: Outcome Relations of *Select Cloud Vendor* 2-3

	Shared Hardware Multi-Tenancy	Shared OS Multi-Tenancy	Shared Middleware Multi-Tenancy	Shared Application Multi-Tenancy	Public Cloud	Private Cloud	Community Cloud	Hybrid Cloud	IaaS	PaaS	SaaS	IaaS + PaaS	IaaS + SaaS	PaaS + SaaS	Iaas + PaaS + SaaS
Select Cloud Vendor	**Select Multi-Tenancy Level**				**Select Cloud Deployment Model**				**Select Cloud Service Model**						
Evaluated Cloud Vendor	eb	eb	eb	eb	eb	eb	eb	eb	eb	eb	eb	eb	eb	eb	eb

Table A.24.: Outcome Relations of *Select Cloud Vendor* 3-3

	On-Premise Hosting	Off-Premise Hosting	Hybrid Hosting	Inhouse	Managmenet	Outsourced	Inhouse + Management	Inhouse + Outsourced	Management + Outsourced	Inhouse + Management + Outsourced	Data In Same Jurisdiction	Data In Different Jurisdiction
Select Cloud Vendor	**Define Cloud Hosting**				**Define Roles of Responsibility**						**Define Resource Location**	
Evaluated Cloud Vendor	eb	eb	eb	eb	eb	eb	eb	eb	eb	eb	eb	eb

A.2. CloudDSF+ Parser

Figure A.1 depicts the implemented classes of the CloudDSF+ Parser in a reduced manner omitting most of the attributes and methods.

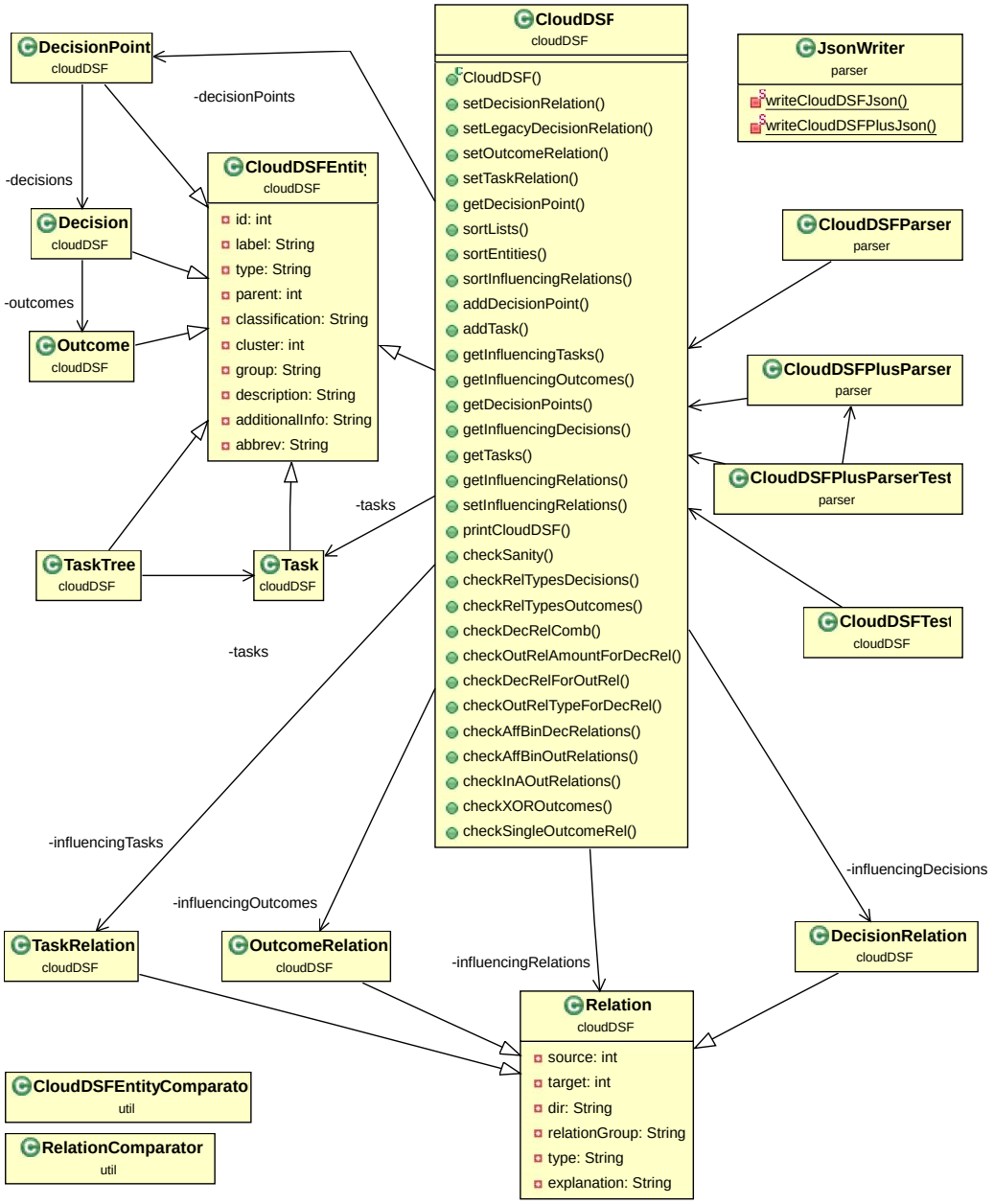

Figure A.1.: Overview of the classes of the CloudDSF+ Parser.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature