# Sharing is Caring
# A Decision Support Model for Multi-Tenant Architectures

*M. Pors*

*L. Blom*

*J. Kabbedijk*

*S. Jansen*

Technical Report UU-CS-2013-015
September 2013

Department of Information and Computing Sciences
Utrecht University, Utrecht, The Netherlands
www.cs.uu.nl

ISSN: 0924-3275

Department of Information and Computing Sciences
Utrecht University
P.O. Box 80.089
3508 TB Utrecht
The Netherlands

# Sharing is Caring
# A Decision Support Model for Multi-Tenant Architectures

Michiel Pors Leen Blom Jaap Kabbedijk Slinger Jansen

**Abstract**

Business software is increasingly moving from a traditional on-premises deployment model to a Software as a Service deployment model. In a Software as a Service deployment model, the possession and ownership of the software application is separated from its use. The software is hosted by a Software as a Service provider, relieving the customer organization from the responsibility for supporting the software, and purchasing and maintaining server hardware for it. The service provider can achieve substantial cost savings by applying economies of scale. This involves a system where customers share services, databases or resources and is known as multi-tenancy.

The option to enable multi-tenancy is not binary. There exist various multi-tenant architectures, because it can be applied at different levels in the architecture. Also, multi-tenancy is not necessarily beneficial, certain situations require a more single-tenant approach. The appropriate level of resource sharing is crucial for a software provider, because it defines an architectural decision. However, there is insufficient knowledge and understanding to determine the most suitable multi-tenant architecture for the software application of a specific Software as a Service provider.

This research focuses on the development of a Multi-Tenant Architecture Selection Model to assist service providers with this architectural decision problem. First by means of a structured literature study a set of twelve multi-tenant architectures is identified. These multi-tenant architectures describe which resources in an application's system are shared among tenants, discriminating between the application and database layer. With the same literature study a list of twenty two decision criteria, representing factors that influence the decision, is identified. They are based on consequences, drawbacks and benefits, considerations and requirements related to multi-tenancy. The multi-tenant architectures and decision criteria are then evaluated by domain experts.

The Analytic Hierarchy Process is selected as the decision making method, based on the complexity of the decision problem, the lack of quantitative data, and the importance of weighing the decision criteria. After this selection, all multi-tenant architectures are ranked on each decision criterion, using domain experts. This results in a decision matrix showing the performance score of each architecture with respect to each criterion. This matrix can then be used by Software as a service providers performing the analytic hierarchy process.

The set of multi-tenant architectures, the list of decision criteria, the decision matrix and the final Multi-Tenant Architecture Selection Model are the key deliverables of this research and support architects in choosing the most suitable architectural pattern. This research is the first step in helping architects of Software as a Service providers make better architectural decisions, saving them time, effort and potential problems in the future.

# 1 Introduction

Business software is increasingly moving from a traditional on-premises deployment model to a Software as a Service (SaaS) deployment model (D. Ma, 2007; W. Sun, Zhang, Chen, Zhang, & Liang, 2007). The traditional solution involves developing software applications that get shipped to the customer to be deployed on-site. This requires clients to own and maintain an in-house IT system with servers running the software.

In this traditional model the clients buy a software license to use the software. This is usually a one-time upfront fee. For line-of-business software this fee potentially includes on-site installation and service visits from the software vendor service teams. These services lead to vendor costs and affects the price at which the software vendor can afford to sell the software application. Therefore, such software is typically targeted at the larger businesses that can afford these expenditures (Chong & Carraro, 2006).

Customers often have their own specific requirements for their software applications. The causes for this variance in customer wishes include: industry focus differences; customer behavior differences; product offering differences; regulation differences; culture differences and operation strategy differences (W. Sun, Zhang, Guo, Sun, & Su, 2008). A software vendor can cater for these varying user wishes by customizing the source code of the software application or give the customer configuration options so the customer can change the application to his liking. Consequently, clients in a traditional on-premises model each have its own modified software application running.

Software as a Service focuses on separating the possession and ownership of a software application from its use (Turner, Budgen, & Brereton, 2003). Within a SaaS environment, software and data is hosted by the software vendor and is delivered online (Dubey & Wagle, 2007). A SaaS customer can access the service provider's applications from various client devices. The applications are running on a cloud infrastructure that is not under the control of the customer (Mell & Grance, 2011). Because the software is hosted by the SaaS provider, the customer organization is relieved from the responsibility for supporting the software, and purchasing and maintaining server hardware for it (Chong & Carraro, 2006).

Furthermore, in a SaaS deployment model, the customer usually does not purchase a software license (W. Sun et al., 2007). Payment is typically based on a subscription revenue model, e.g. the SaaS provider charges his clients per use or on a monthly basis per user (Laplante, Zhang, & Voas, 2008). An example is Salesforce.com, an early leader of the SaaS model, providing on-demand Customer Relations Management and automation tools. Salesforce uses a subscription revenue model and charges clients per user on a monthly basis.

As mentioned before, a SaaS application is hosted by the SaaS provider. Therefore, the customer no longer needs to maintain a large costly in-house server running the software. In addition, the large upfront fee is replaced by a small monthly sum. This enables small and medium-sized businesses to afford otherwise costly business software. SaaS is typically targeted at these type of customers.

One often addressed benefit of SaaS is the ability to apply economy of scale (Sääksjärvi, Lassila, & Nordström, 2005). A SaaS vendor can serve his clients from a centrally-hosted software service. This service, running on the vendor's server, supports multiple clients and enables distributing the server costs over the clients, decreasing the total cost of ownership. Moreover, with each additional client, the individual server costs are reduced. Compared with the traditional model where each client dedicates an entire server to the application substantial cost savings can be achieved.

This aspect of customers sharing servers can be extended to other parts of an application system. For example, customers can share resources such as databases, virtual machines or network connections. This sharing of resources among customers is what in this thesis will be referred to as multi-tenancy.

Multi-tenancy receives increasing attention in scientific literature, but is still a relatively new concept. Therefore, a lot of different definitions can be found and there is yet no single definite description. Two often cited articles in the domain of multi-tenancy are written in 2006 by Chong and Carraro and in 2007 by Guo et al. Chong and Carraro described multi-tenancy as follows: "A SaaS vendor with $x$ number of customers subscribing to a single, centrally-hosted software

service enables the vendor to serve all of its customers in a consolidated environment."(2006, p. 6). This description lacks detail and is more a description of an opportunity in a certain situation. Guo et al. do provide a description more like a definition: "In a multi-tenancy enabled service environment, user requests from different organizations and companies (*tenants*) are served concurrently by one or more hosted application instances based on a shared hardware and software infrastructure."(2007, p. 1). This definition however, states that for a service environment to be multi-tenant, application instances need to be shared. So according to this description service environments in which data tier only resources – like databases – are shared among tenants are not multi-tenant. Because there is disagreement with previous definitions, an other definition of multi-tenancy is used throughout this work. The definition is as follows:

> *Multi-tenancy is a property of a system where multiple varying customers and their end-users share the system's services, applications, databases, or hardware resources, with the aim of lowering costs.*

By using this definition, multi-tenancy can be referred to the sharing of resources in the complete system and not in just a single or a couple of layers or tiers. In addition, multi-tenancy can be viewed as more than just the sharing of application or data instances.

The opposite of multi-tenancy is – not surprisingly – called single-tenancy. In a single-tenant system, no resources are shared among customers. A software solution deployed using the traditional on-premises model is single-tenant.

## 1.1 Problem Statement

Multi-tenancy can entail many benefits. By serving the software service from a centrally hosted location, clients are relieved from the responsibility of purchasing and maintaining big in-house servers. The total cost of ownership decreases and gives the SaaS provider access to new potential customers that previously could not afford the expenses (Chong & Carraro, 2006). In addition, the utilization rate of hardware in a multi-tenant environment is higher than in a single-tenant environment (Sääksjärvi et al., 2005). Furthermore, when multiple customers share application instances and data instances, the total number of instances running will be much lower than in a single-tenant environment. This lower amount of instances is beneficial for maintenance (Kwok, Nguyen, & Lam, 2008) and facilitates application development (Bezemer, Zaidman, Platzbeecker, Hurkmans, & 't Hart, 2010).

However, multiple barriers withhold SaaS providers from massively switching to multi-tenant environments. The challenges of implementing multi-tenancy involve issues with performance (Lin, Sun, Zhao, & Han, 2009), scalability, security (Guo et al., 2007) and re-engineering the current software application (C.-H. Tsai, Ruan, Sahu, Shaikh, & Shin, 2007).

In certain situations multi-tenancy can be very beneficial for a software vendor, but in other circumstances a single-tenant approach will be more suitable. Selecting the appropriate multi-tenant solution is a complex problem, there are many considerations and consequences to take into account. Also, the solution itself is complex, because there exist various multi-tenancy implementations.

Multi-tenancy is defined as a broad concept in this work. There exist multiple degrees of multi-tenancy, described as multi-tenancy levels. This means there exist many configurations of a software system that meet the multi-tenancy definition. Benefits and barriers of multi-tenancy are identified and described in literature, but the aspect of choosing an appropriate multi-tenant architecture based on SaaS providers' preferences has received little attention to date. Finding the most suitable multi-tenant architecture is crucial, because the architecture expresses a fundamental structural organization schema for a provider's software system. However, choosing the best solution is a complex task. Accounting for all the challenges and benefits complicates the decision process considerably.

Related to this problem is a model developed by Kabbedijk and Jansen (2011) depicting deployment solutions that are considered best practices in specific situations. This model includes the

following four deployment solutions: Custom Software Solution, Software Product Line Solution, Standard Multi-tenant Solution and Configurable Multi-tenant Solution.

The model shows the most suitable deployment model is based on the need to share resources and the need to share functionality among customers. It ignores the explicit levels of multi-tenancy and only distinguishes specific situations in the context of the level of resources and the level of functionality shared between tenants. More criteria that influence the decision problem are expected to exist, for example the many barriers of multi-tenancy.

There are currently no other known papers that map multi-tenant architectures to factors advocating or militating different multi-tenant solutions. The formal problem statement for this research project is as follows:

> *The appropriate level of resource sharing is crucial for SaaS providers, because it defines an architectural decision. However, there is insufficient knowledge and understanding to determine the right multi-tenant architecture for the software application of a specific SaaS provider.*

## 1.2 Thesis Outline

This section introduced the reader in the domain of multi-tenancy and described the problem statement this research addresses. Section 2 starts by describing the research objective and research questions. The relevance is then touched, followed by the research design and model. Then, in Section 3, some background knowledge on multi-tenant architectures and decision making theory is provided. Sections 4 to 6 cover research SQ. 1 and SQ. 2. A set of multi-tenant architectures and decision criteria is first identified from literature and then evaluated using experts. The description of the literature study protocol is described in Section 4 and the results thereof are in Section 5. Section 6 covers the evaluation of these results. Then, an explanation for the selection of a specific decision making method is given in Section 7. Next, in Section 8 an answer to the third research subquestion is given. The Multi-Tenant Architecture Selection Model is presented in Section 9. The limitations of this research can be found in Section 10 and the conclusions are defined in Section 11.

# 2 Research Approach

This section starts with describing the research objective. Then, the research questions are showed, followed by the research context. After that, the research design, research process and research model are explained.

## 2.1 Research Objective

Howard (1966, p. 56) was the first to coin the term *decision analysis* on which he said the following:

> Decision analysis is a logical procedure for the balancing of factors that influence the decision. The procedure incorporates uncertainties, values, and preferences in a basic structure that models the decision. Typically, it includes technical, marketing, competitive, and environmental factors. The essence of the procedure is the construction of a structural model of the decision in a form suitable for computation and manipulation.

Decision analysis addresses a decision problem that arises in many industries and business activities in which from a set of possible solutions one must be chosen. The same type of problem is stated in the problem statement of this research. The set of possible solutions then correspond to the various multi-tenant architectures. Solving the decision problem is often difficult, because the decision criteria are usually in conflict with one another. Decision making methods help decision makers to choose among the set of solutions. Examples of areas in which decision making methods are applied include vendor selection, outsource location, layout design, technology investment decisions and engineering problems. This research addresses the problem statement by developing a decision support model, called the Multi-Tenant Architecture Selection Model. For a more detailed description of decision making and the concepts involved, see Section 3.

This developed decision support model should be useful to various SaaS providers. There exists a variety of SaaS providers. The amount of customers, type of application, and domain sector are examples of variables that influence how providers are situated. It describes the setting of those providers. Moreover, these different conditions cause different interests among SaaS providers. The Multi-Tenant Architecture Selection Model should account for this, it should be generic and useful to any SaaS provider.

As is described in more detail in Section 3, decision making is a process consisting of multiple activities in which several artifacts are created. This research carries out some of these activities and develops some of these artifacts. The decision support model should explain what activities remain to be performed with which artifacts by the decision makers.

The Multi-Tenant Architecture Selection Model will have the structure shown in Figure 1. The model illustrates three main phases, each of which consists of a number of steps to be carried out by the decision makers. The first phase, *Assessment*, is responsible for assessing if all the required information is available for the decision makers. If so, the second phase, *Calculation*, can be initiated in which the actual calculation of the most suitable multi-tenant architecture takes place. After this calculation, the final phase, *Architecture Recommendation*, is started in which the decision makers evaluate the result from the previous phase and a recommendation is provided. Apart from the steps, the Multi-Tenant Architecture Selection Model also shows what artifact is to be used in which phase. The exact steps and artifacts are not yet displayed, but the Multi-Tenant Architecture Selection Model will be completed by the end of this research.

## 2.2 Research Questions

Based on the problem statement and the research objective the main research question is formulated as follows:

RQ. *How can a SaaS provider be optimally supported in the decision process of choosing the most suitable multi-tenant architecture?*

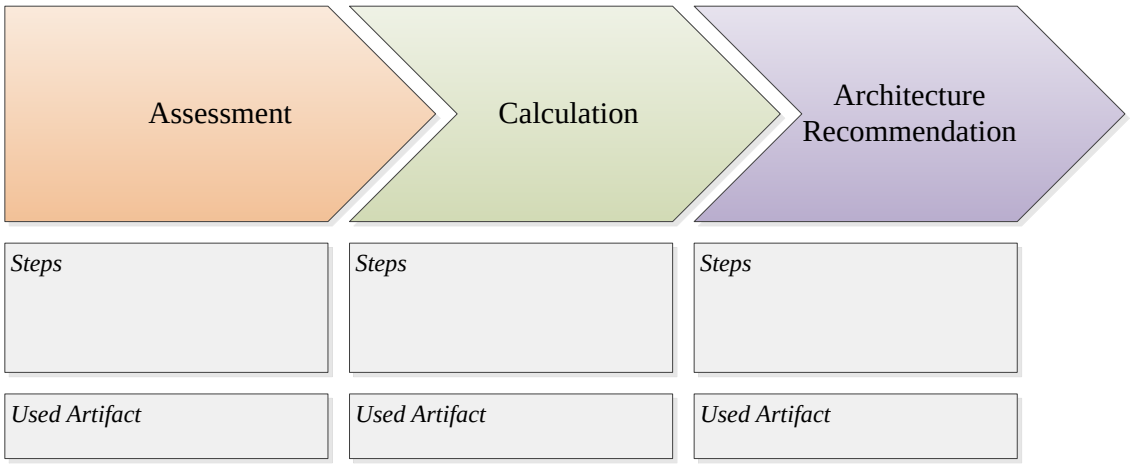



Figure 1: Structure of Multi-Tenant Architecture Selection Model

Several steps need to be carried out in order to develop the Multi-Tenant Architecture Selection Model that solves the main research question. A general decision support model consists of three fundamental elements, all of which need to be identified. The first element is the in decision theory so-called set of alternatives. It corresponds to the various multi-tenant architectures from which a SaaS provider must decide. Hence, the first subquestion is defined as follows:

SQ. 1 *Which multi-tenant architectures currently exist?*

A explained by Howard, a decision is influenced by several factors, the second element to be identified. In decision theory, these factors are called decision criteria or attributes. They discriminate among the alternatives and measure the extent of preference. This leads to the second subquestion:

SQ. 2 *What measurable decision criteria are of importance to SaaS providers in choosing a multi-tenant architecture and define a discrimination among these multi-tenant architectures?*

Finally, the alternatives must be evaluated against the decision criteria, resulting in performance scores. The final subquestion is stated as:

SQ. 3 *How do the multi-tenant architectures perform on each decision criterion?*

## 2.3 Research Context

This section describes the involved stakeholders, and the scope and relevance of this research.

### 2.3.1 Stakeholders

There exist several roles in the realm of software as a service. There are three roles that are of most interest in this research. The SaaS Provider is the entity hosting the service and selling the service. A SaaS Customer uses this service by subscribing to it. Finally, the SaaS Application Developer is the company developing the application that is hosted as a service. A single organization can take on more than one of these roles, for instance a company developing an application and also offering this application as a service to customers.

Other known roles in cloud computing are the infrastructure as a service provider and the platform as a service provider, but this research will focus mainly on the SaaS landscape.

This research use the term SaaS provider and service provider interchangeably. In this work, both a SaaS provider and service provider refer to a company developing its own application and offering it as a service.

### 2.3.2 Scope

Two major categories of software as a service are identified by Chong and Carraro (2006, p. 3):

> **Line-of-business services**, offered to enterprises and organizations of all sizes. Line-of-business services are often large, customizable business solutions aimed at facilitating business processes such as finances, supply-chain management, and customer relations. These services are typically sold to customers on a subscription-basis.
>
> **Consumer-oriented services**, offered to the general public. Consumer-oriented services are sometimes sold on a subscription-basis, but are often provided to consumers at no cost, and are supported by advertising.

This research focuses on line-of-business solutions. The multi-tenant architectures and criteria identified are based on that context. Some concepts however, might apply to the scope of consumer-oriented services as well. In addition, the multi-tenant architectures discussed in this work are structured for software applications using the layered services pattern. This pattern separates concerns by logically isolating each layer (Fowler, 2002). Also, the architectures are structured for systems with a tiered distribution in mind. This means the layers are physically separated and each tier addresses one or more layers (Fowler, 2002). Thus, the multi-tenant architectures in this work are focused on software applications using the layered services pattern in combination with the tiered distribution pattern. Nevertheless, the Multi-Tenant Architecture Selection Model can be of interest for service providers offering applications that lack the tiered distribution pattern.

### 2.3.3 Scientific Relevance

Multi-tenancy is an important concept in system architecture and receives increasingly more attention in scientific literature. Most of this literature is only focused on problems in *native* or *full* multi-tenancy, i.e. in which a single application instance is offered to multiple tenants. There are different types of multi-tenancy however, and the question of how and where to apply multi-tenancy remained hitherto neglected. This research fills this gap in literature with the development of a decision support model by means of decision making theory.

Furthermore, most research on multi-tenancy is focused on applying multi-tenancy at one tier or level only. In contrast, this research takes a holistic view at multi-tenant architectures and not just one layer or tier. The knowledge base of scientific literature on multi-tenancy is increased with the structuring of generic multi-tenant architectures. These architectures cover both the data and application layer.

### 2.3.4 Social Relevance

The practical relevance of this research is represented by the key deliverable: the Multi-Tenant Architecture Selection Model. It can be used by SaaS providers struggling how to structure the system architecture for their software services and to select the components to share among customers. When used by a SaaS provider, the decision support model describes what activities to execute, resulting in one or more multi-tenant architectures the provider can select or apply deeper analysis on.

In addition to the result of the Multi-Tenant Architecture Selection Model itself, the decision making process is also useful. It provides decision makers with insight how their decision criteria relate to each other and which criteria are considered more important.

## 2.4 Research Design

According to Hevner, March, Park, and Ram (2004) there are two paradigms characterizing the research in the Information Systems discipline, these are behavioral science and design science. The behavioral science paradigm addresses the development and verification of theories about human or organizational behavior. The design science paradigm aims to extend the knowledge

base with the development of new artifacts, solving important and relevant business problems. The objective of this research is the development of a decision support model, it represents an artifact solving the business problem stated in the problem definition. Therefore, this research is defined as design-science research.

Peffers et al. (2006) developed a Design Science Research Process (DSRP) model. This model shows the process elements that are present in design-science research. It is based on seven papers and presentations discussing components of the design-science research process (Archer, 1984; Takeda, Veerkamp, & Yoshikawa, 1990; Nunamaker Jr & Chen, 1990; Eekels & Roozenburg, 1991; Walls, Widmeyer, & El Sawy, 1992; Rossi & Sein, 2003; Hevner et al., 2004). The DSRP model is depicted in Figure 2.

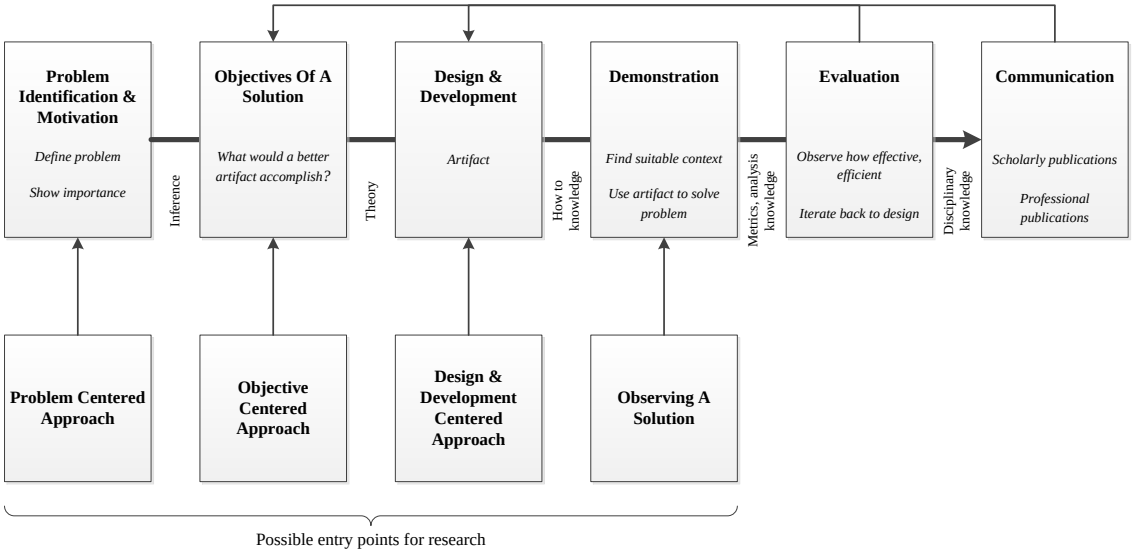



Figure 2: Design Science Research Process Model by Peffers et al. (2006)

The process model consists of six activities in a nominal sequence: *Problem Identification & Motivation*, *Objectives Of A Solution*, *Design & Development*, *Demonstration*, *Evaluation*, and *Communication*. The model also depicts possible entry points for research approaches. Activities one to four can be start positions for these different research approaches and move from there. This research has a problem-centered approach, therefore this research starts activity 1, *Problem Identification & Motivation*. The problem definition and motivation are described in Section 1.1. The second activity, which covers the objective of the solution, is described in Section 2.1. The third activity, responsible for the design and development of the artifact, is described in the next section. Due to time constraints, this research omits the execution of the fourth and fifth activity, i.e. demonstrating and subsequently evaluating the artifact developed in the third activity. It is suggested these activities are of subject in further research. Finally, the deliverable of the sixth activity, for the communication of the performed activities, is this written thesis.

## 2.5 Research Process

The process of constructing the Multi-Tenant Architecture Selection Model is illustrated in Figure 3 and consists of three core activities. First, a structured literature study is carried out to identify both the multi-tenant architectures and the decision criteria. Then, these two artifacts are evaluated with the aid of domain experts. In the final core activity a questionnaire is conducted with domain experts to construct a decision matrix showing the performance values of the multi-tenant architectures with respect to each decision criterion. The process model is explained

in more detail in the following section.

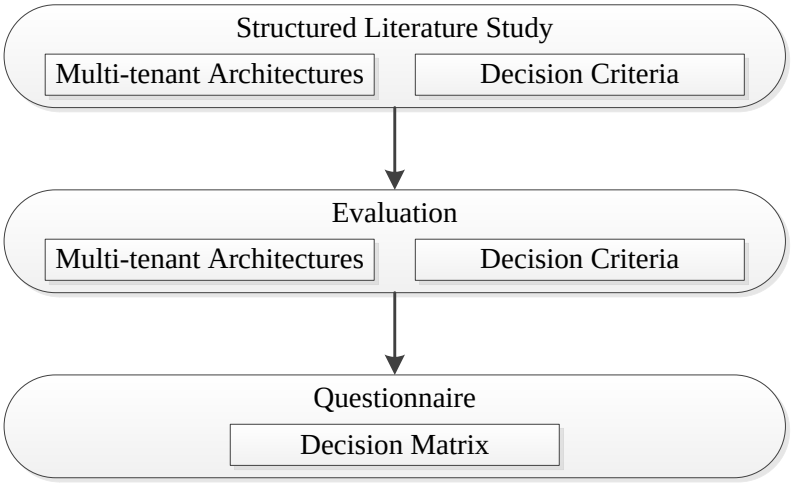



Figure 3: Research Process

## 2.6 Research Model

A more detailed research model is explained using a process deliverable diagram (PDD), see Figure 4. This type of diagram, as described by van de Weerd and Brinkkemper (2009), consists of a meta-process model and a meta-data model that are connected with each other. The left side shows the meta-process model and displays the various activities and the sequence of their execution. The right side depicts the meta-data model and shows the deliverables and their relations. Each deliverable is linked with the activity that produced it. A PDD is always accompanied by an activity table and a concept table. The activity table lists the activities and describes them in more detail. The same applies for the concept table: each deliverable is listed and described. The activity and concept table of Figure 4 are shown in Tables 1 and 2.

The PDD consists of five phases, named *Multi-tenant Architectures & Criteria Identification*, *Evaluation*, *Decision Making Method Selection*, *Performance Identification*, and *Decision Support Model Construction*. The phases are visualized as gray-colored round edge rectangles covering related research activities. The following sections describe each phase in more detail.

### 2.6.1 Multi-tenant Architectures & Criteria Identification

The first two phases answer the fist two subquestions. The first subquestion is responsible for finding existing multi-tenant architectures. As mentioned earlier, the Multi-Tenant Architecture Selection Model should be generic. This means the model should encompass a set of generic multi-tenant architectures. This set should include all possible multi-tenant architectures, but a decision support model can only include a certain number of solutions. So, a balance must be found between a set of architectures that defines all possible multi-tenant approaches and a set that is useful in a decision support model.

By means of a literature study, these multi-tenant architectures are identified. Instead of searching directly for multi-tenant architectures and generalize those, a different approach is taken. First, levels and tiers on which multi-tenancy can be applied are identified. Then, generic multi-tenant architectures are formed from these levels and tiers. The architectures used in the Multi-Tenant Architecture Selection Model are represented by visual renderings displaying the arrangement of shared and non-shared resources among tenants. This approach is considered less time-consuming. The literature study protocol is described in detail in Section 4. The results of

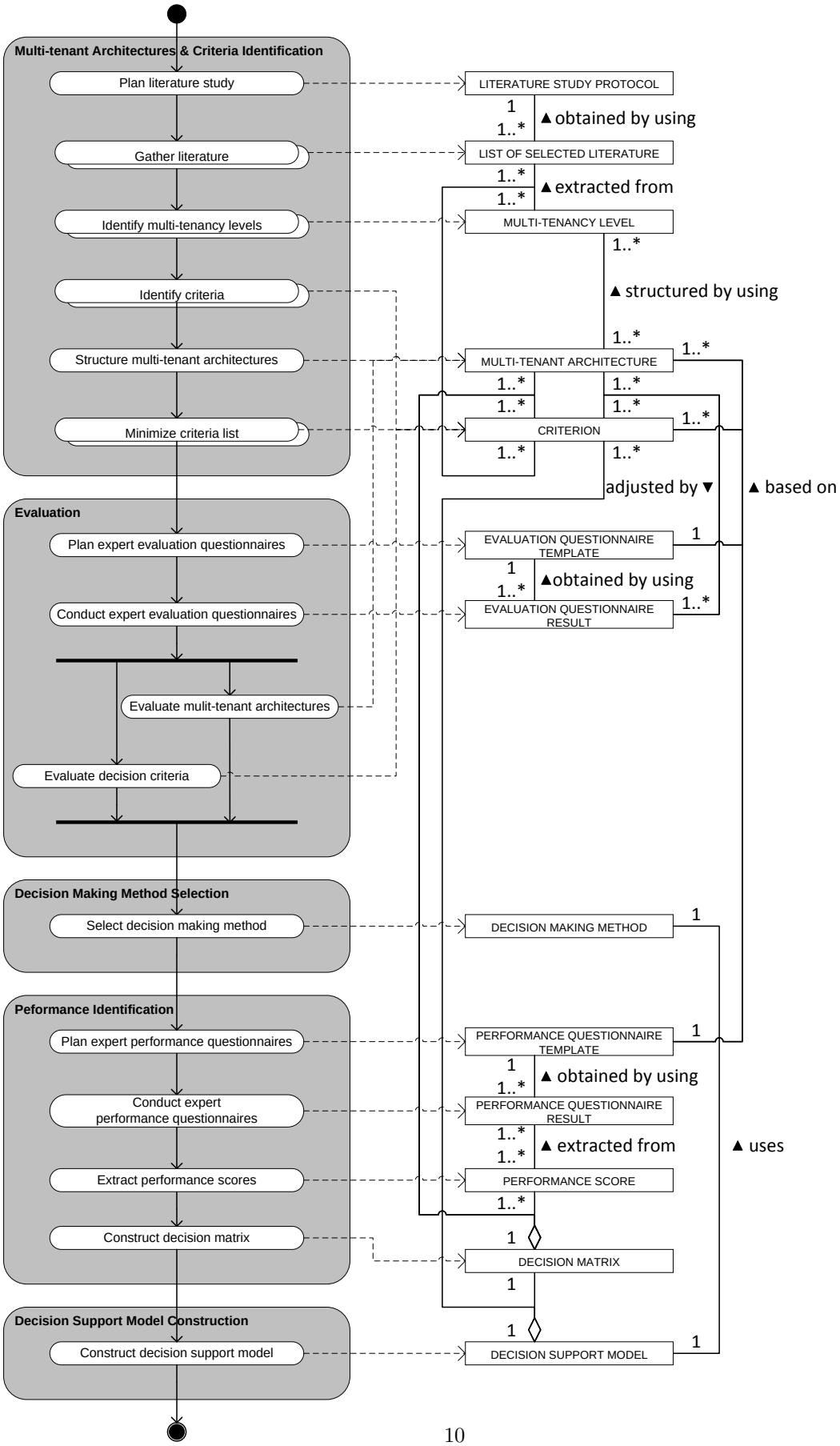





Figure 4: Research Model

Table 1: Research Model Activity Table

| Activity | Sub-Activity | Description |
|---|---|---|
| Multi-tenant Architectures & Criteria Identification | Plan literature study | A LITERATURE STUDY PROTOCOL is composed which describes how the literature review is conducted. Guidelines from literature on how to write this are followed. |
| | Gather literature | After constructing the protocol, the collection of scientific literature is started. Multiple libraries are searched and inclusion and exclusion criteria is used to filter the initially large set of articles. This results in a LIST OF SELECTED LITERATURE. |
| | Identify multi-tenancy levels | The LIST OF SELECTED LITERATURE is searched for various degrees of multi-tenancy. Multi-tenancy can be applied at different layers or tiers which requires different approaches. This activity results in a set of MULTI-TENANCY LEVELS. |
| | Identify criteria | From the same LIST OF SELECTED LITERATURE, criteria are searched for at locations where MULTI-TENANCY LEVELS are identified. This leads to a set of CRITERIA. |
| | Structure multi-tenant architectures | The various MULTI-TENANCY LEVELS are structured in a set of MULTI-TENANT ARCHITECTURES. |
| | Minimize criteria list | The initially large list of CRITERIA is minimized by merging equal-meaning criteria and deleting infrequent ones. |
| Evaluation | Plan evaluation questionnaires | A series of evaluation meetings is planned to evaluate the constructed MULTI-TENANT ARCHITECTURES and identified CRITERIA with experts. An EVALUATION QUESTIONNAIRE TEMPLATE is composed. |
| | Conduct evaluation questionnaires | The questionnaires are administered with individual experts. This results in a set of EVALUATION QUESTIONNAIRE RESULTS. |
| | Evaluate multi-tenant architectures | After all questionnaires, the EVALUATION QUESTIONNAIRE RESULTS are used to analyze if each multi-tenant architecture should be incorporated in the DECISION SUPPORT MODEL. |
| | Evaluate decision criteria | As with the previous activity, the EVALUATION INTERVIEW RESULTS are used to analyze if each criterion should be incorporated in the DECISION SUPPORT MODEL. |
| Decision Making Method Selection | Select decision making method | A specific decision making method is chosen based on the characteristics and features of that method. It results in a DECISION MAKING METHOD. |
| Performance Identification | Plan performance questionnaires | A series of meetings is planned to obtain scores for the constructed MULTI-TENANT ARCHITECTURES with respect to the identified CRITERIA with experts. A PERFORMANCE QUESTIONNAIRE TEMPLATE is composed. |
| | Conduct performance interviews | The questionnaires are administered with individual experts. They result in a set of PERFORMANCE QUESTIONNAIRE RESULTS. |
| | Extract performance scores | The individual scores provided by the experts are aggregated to obtain PERFORMANCES SCORES for all multi-tenant architectures on each decision criterion. |
| | Construct decision matrix | The PERFORMANCE SCORES are combined with the MULTI-TENANT ARCHITECTURES and CRITERIA resulting in a DECISION MATRIX. |
| Decision Support Model Construction | Construct decision support model | The actual Multi-Tenant Architecture Selection Model is constructed. |

Table 2: Research Model Concept Table

| Concept | Definition |
|---|---|
| LITERATURE STUDY PROTOCOL | A document describing how the literature study is conducted. It includes a search strategy with the used data sources and search terms; a description of the study selection criteria; a procedure for the study selection; and a strategy for data extraction and analysis. It is described in Section 4. |
| LIST OF SELECTED LITERATURE | A list of scientific articles remaining after applying inclusion and exclusion criteria. The full texts of these papers are obtained. |
| MULTI-TENANCY LEVEL | A certain level, layer, or tier on which multi-tenancy can be applied, or a degree or approach which involves multi-tenancy. They are shown in Section 5.2.1. |
| MULTI-TENANT ARCHITECTURE | An arrangement of elements displaying which resources of an application's system are shared among tenants. They are listed in Figures 10 to 21. |
| CRITERION | A measurable attribute discriminating among the MULTI-TENANT ARCHITECTURES. It defines an important factor for a SaaS provider to evaluate the MULTI-TENANT ARCHITECTURES against. They are described in Table 16. |
| EVALUATION QUESTIONNAIRE TEMPLATE | A paper form containing evaluation questions for the MULTI-TENANT ARCHITECTURES and CRITERIA. It is used in the evaluation questionnaire. It is included in Appendix D. |
| EVALUATION QUESTIONNAIRE RESULT | An evaluation questionnaire completed by an expert. The data of the paper questionnaire is transfered to statistical analysis software and used to evaluate the MULTI-TENANT ARCHITECTURES and CRITERIA. See Tables 17 to 18 for the results. |
| DECISION MAKING METHOD | A technique for organizing and analyzing a decision problem. It usually consists of a series of steps in order to perform the method. There exists different schools of thought for solving decision problems, each with its own pros and cons. The selection and result is described in Section 7.5. |
| PERFORMANCE QUESTIONNAIRE TEMPLATE | A paper form containing questions to rate the performance of the MULTI-TENANT ARCHITECTURES against the CRITERIA. It is used to conduct the performance questionnaires. The template is in Appendix E. |
| PERFORMANCE QUESTIONNAIRE RESULT | A ratings questionnaire completed by an expert. The data of the paper questionnaire is transfered to statistical analysis software and used to calculate the performance of the MULTI-TENANT ARCHITECTURES against the CRITERIA. Results can be found in Appendix F. |
| PERFORMANCE SCORE | A value representing the performance of a multi-tenant architecture with respect to a criterion. It is an aggregated number based on a set of values provided by several experts. A higher value means a higher performance. |
| DECISION MATRIX | A matrix displaying the PERFORMANCE SCORES of each MULTI-TENANT ARCHITECTURE with respect to the CRITERIA. It is shown in Table 19. |
| DECISION SUPPORT MODEL | A process model showing the steps that need to be carried out and the artifacts that need to be used. It makes use of the selected DECISION MAKING METHOD. It is described in detail in Section 9 and the main deliverable of this research. |

this literature study is the subject in Section 5. To ensure the formed architectures represent feasible architectures, they are evaluated. This evaluation is done in the second phase of the research model.

The second research subquestion is responsible for identifying decision criteria. These criteria are measurable attributes that discriminate among the multi-tenant architectures. For a more detailed description, read Section 3.2.2. The literature study in the first phase of the research model is also carried out to identify the decision criteria. This identification process results in a large set of criteria. Prior to evaluating this set, which is done in conjunction with the multi-tenant architectures in the second phase, the list of criteria is analyzed to merge similar and delete unimportant attributes.

### 2.6.2 Evaluation

The evaluation of the multi-tenant architectures and decision criteria is conducted using a questionnaire. With this survey a dozen of experts are asked for their opinions on the structured

multi-tenant architectures and the composed set of decision criteria. These two elements are subsequently adjusted based on this evaluation. The overall evaluation process is described in Section 6.

### 2.6.3 Decision Making Method Selection

After the evaluation, a decision making method is selected. This comprises the third phase of the research model. There exist many different decision making methods and each method contains pros and cons. This selection is, inter alia, based on the complexity of the decision problem, the cognitive load of the method, its support for qualitative or quantitative data, and its support for group decision making. The selection of a suitable decision making method is discussed at length in Section 7.5.

### 2.6.4 Performance Identification

The final research subquestion, responsible for assessing the performance of the multi-tenant architectures with respect to each decision criterion, is solved in the fourth phase. This is carried out by means of a questionnaire completed by several experts and is covered by Section 8. After extraction, the performance values are combined with the decision criteria and multi-tenant architectures in a decision matrix.

### 2.6.5 Decision Support Model Construction

As all key deliverables are developed, the actual Multi-Tenant Architecture Selection Model can be constructed. This is done in the final phase. It is described in Section 9.

# 3 Theoretical Background

This section elaborates on the concept multi-tenancy and explains decision making theory in more detail. If the reader considers himself comfortable with one of these topics, the corresponding section can be skipped.

## 3.1 Multi-Tenancy

Multi-tenancy in the realm of hardware and software systems features a relative recent attention in scientific literature with the first notion of the term in a paper on the MSDN Library by Chong and Carraro (2006). There is yet no well-established formal definition for multi-tenancy, but scientific literature agrees multi-tenancy is about consolidating multiple customers on a shared operational environment (Chong & Carraro, 2006; Guo et al., 2007; Jacobs & Aulbach, 2007). The hardware and software infrastructure is shared in such an environment, and a hosted application can serve user requests from multiple companies concurrently (Guo et al., 2007).

Multi-tenancy is also regarded as a key attribute of well-designed SaaS applications (Chong & Carraro, 2006). Chong and Carraro developed a commonly-used maturity model of SaaS that distinguishes a total four maturity levels. The last two maturity levels in this model contain multi-tenancy, rendering it as a requirement for a matured SaaS application. The National Institute of Standards and Technology (NIST) regards multi-tenancy as an essential attribute of cloud computing (Mell & Grance, 2011).

Multi-tenancy is not confined to specific resources, but applicable at different levels in a system's architecture. As a result, various approaches to a multi-tenant architecture are possible (Osipov, Goldszmidt, Taylor, & Poddar, 2009; Natis & Knipp, 2008). Each approach entails certain benefits and drawbacks and the graphical representation of such an architecture should explain where and to what extent multi-tenancy is applied. Part of this research is responsible for identifying all these relevant multi-tenant architectures.

## 3.2 Decision Making

In 2005, Figueira et al. explained that a decision can be related to a plurality of point of views, which can be defined as criteria. The approach of accounting for pros and cons of a plurality of point of views is the domain of Multiple Criteria Decision Making (MCDM). MCDM can be divided in two major areas: Multiple Attribute Decision Analysis (MADM) and Multiple Objective Decision Analysis (MODM) (Zimmermann, 1991). In MODM problems, the best alternative is designed based on the given conflicting objectives (Hwang & Masud, 1979). With MADM the alternatives are already available and a preference is based on the conflicting attributes of the alternatives (Hwang & Yoon, 1981).

### 3.2.1 Multi-Attribute Decision Making

This research addresses a problem that can be solved by using MADM. Multi-tenant architectures are already described on a high-level in literature, i.e. the alternatives are available. Therefore, this research will use a decision making technique based on MADM. Problems addressed with MADM can be very different from one another, but they all share the following characteristics (Yoon & Hwang, 1995):

**Alternatives** A finite number of alternatives that offer different approaches. They are screened, prioritized, selected, and/or ranked. Other terms used, among others, are option, policy, action, solution, and candidate.

**Multiple Attributes** Multiple attributes that discriminate among the alternatives. They define measures how well the alternatives achieve preferences of the decision makers. The amount of

attributes depends on the problem setting. When the number of attributes is large, they can be arranged in a hierarchical manner. The term criteria is frequently used too.

**Incommensurable Units** The attributes can have different units of measurements.

**Attribute Weights** Defines the relative importance of the attributes, usually according to an ordinal or cardinal scale. They can be directly assigned by the decision maker or developed using certain weighing methods.

**Decision Matrix** Expresses the problem in a matrix format, where columns represent the alternatives and rows the multiple attributes. The elements of the matrix indicates the performance score of the corresponding alternative against the corresponding attribute. Table 3 is an example of a decision matrix:

Table 3: Typical Decision Matrix

| Criteria | Weights | Alternatives | | | |
|---|---|---|---|---|---|
| | | $A_1$ | $A_2$ | $\cdots$ | $A_N$ |
| $C_1$ | $W_1$ | $a_{1,1}$ | $a_{1,2}$ | $\cdots$ | $a_{1,N}$ |
| $C_2$ | $W_2$ | $a_{2,1}$ | $a_{2,2}$ | $\cdots$ | $a_{2,N}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ |
| $C_M$ | $W_M$ | $a_{M,1}$ | $a_{M,2}$ | $\cdots$ | $a_{M,N}$ |

where:

$C = \{C_i, \text{ for } i = 1, 2, ..., M\}$ is a finite set of $M$ criteria,
$A = \{A_j, \text{ for } j = 1, 2, ..., N\}$ is a finite set of $N$ alternatives,
$W_i$ is the weight belonging to criteria $C_i$,
$a_{i,j}$ represents the performance value of alternative $A_j$ on criterion $C_i$.

Usually the performance values are described using maximization. This means that if $a_{i,k} > a_{i,l}$, then $a_{i,k}$ performs better than $a_{i,l}$ on criterion $C_i$. Minimization is the opposite.

### 3.2.2 Attributes' Properties

According to Keeney and Raiffa (1993) there are five principles the criteria set should meet. The set of attributes should be *complete, operational, decomposable, non-redundant*, and *minimal*. These principles are explained below.

**Completeness** First, the set should be complete. This means that all important aspects of the problem are covered by the attributes.

**Operational** Another recommended property is that the set be operational. In other words, the set of attributes should be useful to help the decision maker choose the best option. They should be meaningful and understandable.

**Decomposable** In addition, it is desired that the set is decomposable. The number of attributes defines the dimension of the decision problem. Solving decision problems involves assessments that get more complex for higher dimensions. This complexity can be restrained by decomposing the assessments in multiple parts of lower dimensions.

**Non-redundancy** Furthermore, the set should contain no redundancies. By having no redundancies in the set, double calculations are avoided.

**Minimum size** Finally, it is desirable to keep the set of attributes as small as possible. The higher the amount of attributes, the harder it becomes to obtain attribute preferences and joint probability distributions.

### 3.2.3 General Decision Making Process

Baker et al. (2001) provides a step by step process applicable for solving any decision problem. In shows in what order the characteristics and elements of a decision making process should be identified. This process is illustrated in Figure 5 using a meta-process model, which is the left hand side of a process-deliverable diagram.

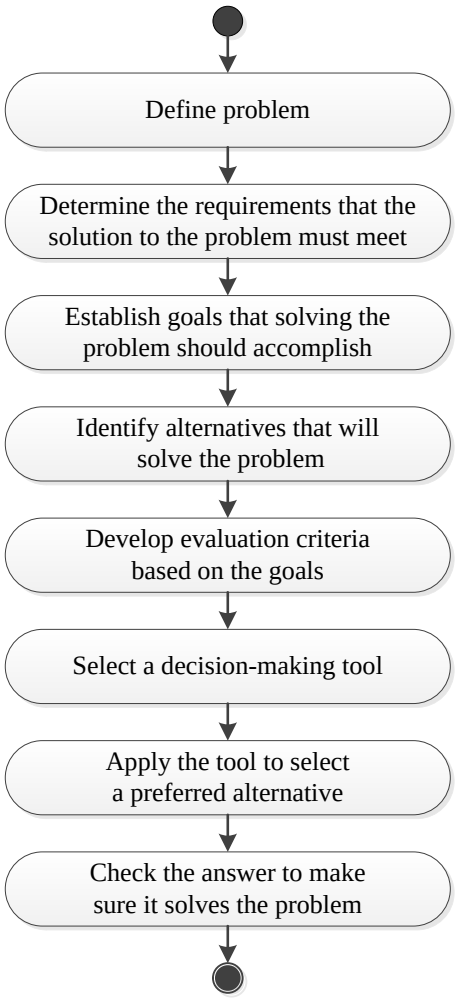



Figure 5: Decision Making Process by Baker et al. (2001)

The first step is equivalent to the first activity in the Design Science Research Process Model in Figure 2 in which by analyzing conditions and identifying causes a clear problem statement is formed (Baker et al., 2001).

The next two steps cover the determination and gathering of requirements and goals respectively. Requirements are conditions the alternatives must meet. Goals represent desirable statements for solutions to have or do and may conflict. Both requirements and goals are provided by

experts in disciplines like operations, maintenance, safety etc. in concurrence with the decision makers. They are normally developed prior to identifying alternatives, because solutions not meeting these requirements or goals can be discarded. The Multi-Tenant Architecture Selection Model developed in this work, however, should be generic and useful to any service provider. Therefore, no alternatives should be discarded beforehand and requirements and goals should be defined and set by each SaaS provider individually upon using the decision model.

Next is the identification of the alternatives. They represent different approaches to solve the problem statement and usually vary in their extent of meeting requirements and goals. A written description and graphical explanation can help to clarify the workings of the alternative and how it differs from the other alternatives. Then, the evaluation criteria, or attributes, can be identified. They should be based on the goals and discriminate among the alternatives. Thereafter a decision making method is to be selected so the alternatives can be evaluated against the decision criteria. Finally, after the selection of a preferred alternative, it should be validated to check if it in fact solves the problem. Also, it should meet the requirements and goals.

This research follows the decision making process. All the activities are therefore in certain way addressed in this research. The definition of the decision problem, representing the first activity in the decision making process, is described in the problem statement of this research. Activity two and three are covered in the first phase of the Multi-Tenant Architecture Selection Model, see Figure 1. The identification of the alternatives and criteria, representing the fourth and fifth activity, is addressed in the first and second phase of the research model. Activity four and five also correspond to SQ. 1 and SQ. 2, respectively. The sixth activity, dealing with the selection of a decision method, is the subject of the third phase of the research model. Finally, the second-to-last and last activity correspond to the second and third phase of the Multi-Tenant Architecture Selection Model, respectively.

# 4 Literature Study Protocol on Multi-tenancy Levels and Decision Criteria

Research SQ. 1 and SQ. 2 are partly tackled by means of a structured literature study. The rationale for this literature study is to identify multi-tenant architectures and decision criteria, which discriminate among these architectures, using an unbiased search strategy. In addition, it is necessary to identify possible existing literature reviews on the topic of multi-tenancy.

This review follows the guidelines of Kitchenham and Charters (2007). Accordingly, first a literature review protocol is created, adhering to the first review phase stated by Kitchenham and Charters. A literature review protocol describes the method that will be used to conduct the systematic review and helps to reduce researcher bias. It consists of the following components:

- Background;
- Research Questions;
- Search Strategy;
- Study Selection Criteria;
- Study Selection Procedures;
- Study Quality Assessment;
- Data Extraction Strategy;
- Data Analysis Strategy;
- Dissemination Strategy;
- Project Timetable;

The study quality assessment can be undertaken to provide even more detailed inclusion and exclusion criteria (Kitchenham & Charters, 2007). This was not necessary for this research and is therefore omitted from the literature review protocol. The dissemination strategy is irrelevant, because the literature review report is part of this thesis. The background and addressed research questions are already discussed in this research, in Section 1.1 and in Section 2.2, respectively. The findings of this literature review are provided in Section 5. A project timetable is created in the internal protocol, but is irrelevant for this thesis and therefore omitted. All other components are described in this section.

## 4.1 Search Strategy

According to Kitchenham and Charters (2007), a search strategy is necessary so readers can assess the rigor of the strategy and the completeness and repeatability of the process. A search strategy includes the resources and search terms to be searched. Preliminary searches are conducted to identify possible existing systematic reviews on the topic of multi-tenant architectures. No already existing systematic reviews on multi-tenant architectures are found.

### 4.1.1 Data Sources

The search process is an electronic search in digital libraries. The selected libraries are the following:

- ACM Portal
- IEEE Xplore Digital Library
- ScienceDirect
- SpringerLink
- Scopus

The first three digital libraries are selected based on their relevance on the topic of Software Engineering (Brereton, Kitchenham, Budgen, Turner, & Khalil, 2007). SpringerLink and Scopus are mentioned by Kitchenham and Charters (2007) as good libraries on Software Engineering and are selected for this reason. SciVerse Scopus' content covers, inter alia, journals provided by publishers like IEEE and Springer. Scopus' database is therefore searched last and already found articles from searches in the other databases are filtered out. The digital libraries have slightly different practices for search commands. The search string is modified to accommodate this.

### 4.1.2 Search Terms

The search terms used are derived from SQ. 1. A list of synonyms, abbreviations and alternative spellings is drawn up by breaking up the research question into individual facets. The search strings are then constructed using the Booleans AND and OR. The reason only SQ. 1 is used to derive the search string, is because research SQ. 2 and SQ. 3 refer to the facets in the first subquestion. From SQ. 1 the following facets are derived:

- multi-tenant
- architecture

The term *"software as a service"* is added to clarify literature focusing on multi-tenant architectures of software applications is seeked. The following synonyms and alternative spellings are drawn up for these concepts:

- tenancy, tenant
- architecture, architectural
- software, service, application, SaaS, software as a service

Trial searches are conducted to examine how the search process can be carried out and how the search string can best be constructed. The search strings are based on various combinations of:

- multi-tenancy, multi-tenant
- software, service, application, architecture

The facet *"software as a service"* is covered by the terms *software* and *service* and is therefore not used. Using search strings with search terms in phrases surrounded by quotation marks, e.g. "multi-tenant architecture", resulted in the exclusion of some already known primary studies. On the other hand, separate use of the search terms, e.g. (multi-tenancy OR multi-tenant) AND (software OR architecture), led to too many results.

New trial searches showed that studies identifying multi-tenant architectures not always addressed these in their abstracts. Therefore, searching for the mentioned search terms in abstracts did not result in all relevant papers. But using all search terms in full text ended up in too much literature. For that reason, literature is first selected based on their relevance on multi-tenancy. This is carried out by a search using the article's abstracts and keywords. The following search terms are used:

- tenan* OR multitenan*

- software OR service OR application OR saas

An asterisk is used as a wild-card and represents variations of the corresponding word, e.g. *tenan** represents *tenant* and *tenancy*. The top line is used for the article's abstract and keywords, the bottom line is only used for the abstract. The search string is constructed by linking the two OR lists using the Boolean AND. This results in the following generic search string:

> ABSTRACT:((tenan* OR multitenan*) AND (software OR service OR application OR saas)) AND KEYWORDS:(tenan* OR multitenan*)

## 4.2 Study Selection Criteria

Study selection criteria assess the relevance of the literature found in the first step. The selection criteria are piloted on a subset of primary studies. The initial electronic search results in a large number of totally irrelevant papers, and using these criteria a smaller, more relevant list of literature can be created. The following criteria are used.

### Inclusion Criteria

1. any article focusing on the topic of multi-tenancy in a hardware or software environment.
2. any article that is cited by other literature in the description of multi-tenancy levels.

### Exclusion Criteria

1. articles that don't appear in scientific papers or conference proceedings.
2. articles already obtained by other digital libraries.
3. articles written in a different language than English.
4. articles of which no full copy can be obtained.
5. articles with no description of multi-tenancy levels.
6. articles in which no prospective decision criteria can be found.

## 4.3 Study Selection Procedures

The study selection procedure and data collection procedure of this literature study is depicted in Figure 6 using a PDD. The accompanying activity table and concept table are represented in Tables 4 and 5, respectively. Three phases can be identified:

1. Literature Gathering
2. Multi-Tenancy Levels Identification
3. Criteria Identification

### 4.3.1 Literature Gathering

The first phase deals with the gathering of the relevant scientific articles. The first process in this phase involves conducting the search using the previous mentioned search string on the selected digital scientific libraries and results in a LIST OF ALL LITERATURE on the topic of multi-tenancy. The next process in this phase is applying the first inclusion criterion and exclusion criteria one to four. All study selection criteria are applied by a single researcher. The rejected papers are maintained in a list with the reasons for exclusion. The accepted articles form the LIST OF CANDIDATE LITERATURE. Next, representing the third process *Perform keyword query*, another selection is carried out, again by a single researcher. This selection involves a keyword query on the full texts of the studies in the LIST OF CANDIDATE LITERATURE. The rationale for this keyword query is mainly simplifying the keyword scanning process rather than reducing the LIST OF CANDIDATE LITERATURE. This query is carried out using qualitative data analysis software and highlights the keywords found in the full texts. The following keywords are searched in this query:

- pattern
- type
- level
- degree
- approach

This query results in a LIST OF RELEVANT LITERATURE with highlighted keywords.

The second and third phase consist of activities that focus on a single study from the LIST OF RELEVANT LITERATURE. The final conditional branch in Figure 6 clarifies this. When each candidate study has been subjected to the activities in the second and third process, the procedure is finished.

### 4.3.2 Multi-Tenancy Levels Identification

The second phase deals with the second inclusion and the fifth exclusion criterion. During the first process in the second phase, *Scan for multi-tenancy levels*, the highlighted keywords are scanned for possible multi-tenancy levels. If found, the follow-up process is to check if there exist citations at the mentioned multi-tenancy levels. If not, the source is included in the LIST OF SELECTED LITERATURE ON MULTI-TENANCY LEVELS. If there are citations, the multi-tenancy levels are searched for in that cited source. There is a loop to handle situations when a cited article itself is citing other sources when describing multi-tenancy levels. Only the final original sources get included in the LIST OF SELECTED LITERATURE ON MULTI-TENANCY LEVELS.

### 4.3.3 Criteria Identification

The third phase is responsible for the sixth exclusion criterion and the identification of decision criteria. As described in Section 3.2.2, the set of attributes should have five properties: *complete*, *operational*, *decomposable*, *non-redundant*, and *minimal*. To meet the *complete* principle, the whole LIST OF RELEVANT LITERATURE is searched for possible decision criteria. This way, the examined

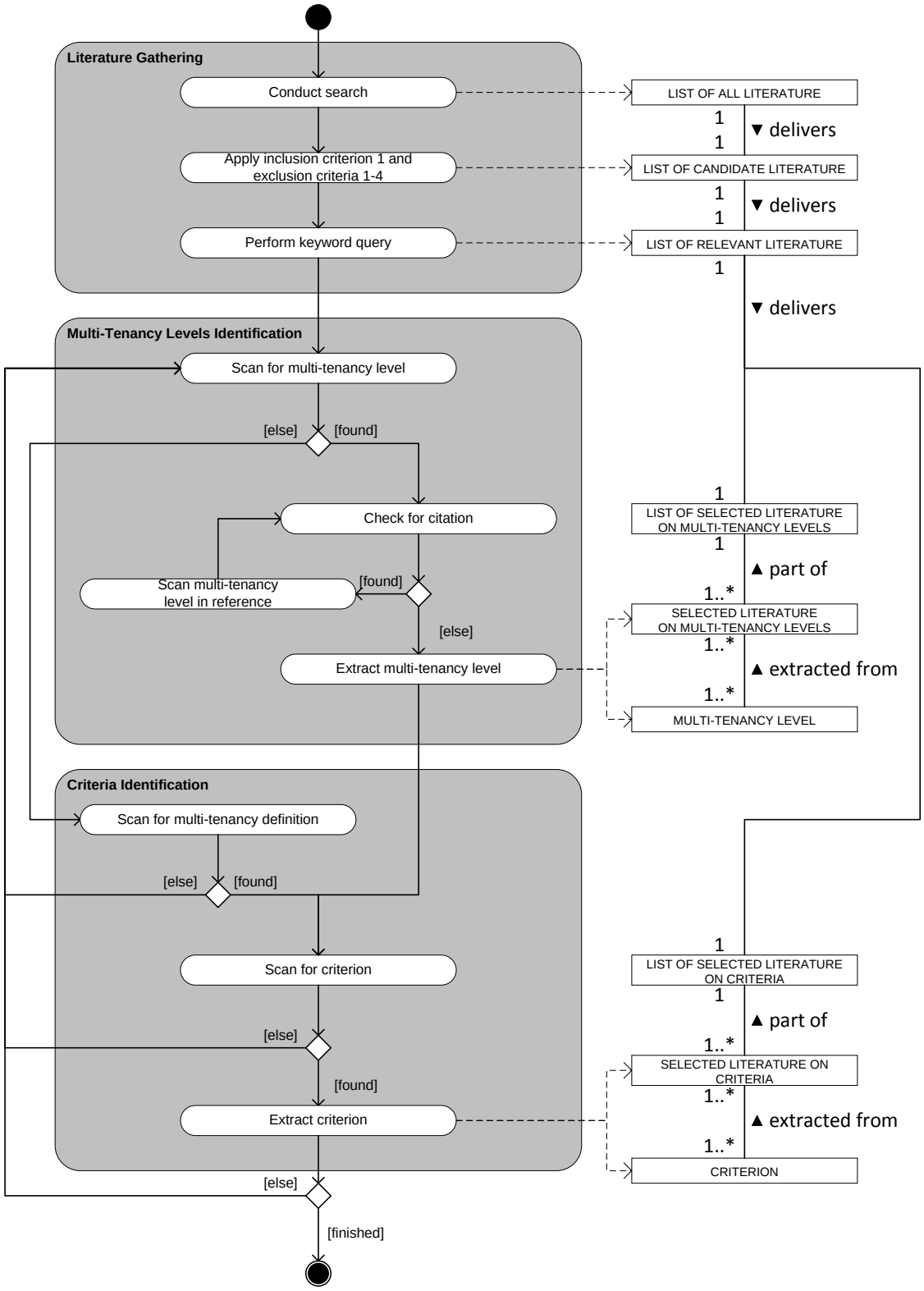



Figure 6: PDD for the Identification of the Multi-Tenancy Levels and Criteria

Table 4: MT Levels and Criteria Identification Activity Table

| Activity | Sub-Activity | Description |
|---|---|---|
| Papers Gathering | Conduct search | The defined search query is used on the selected scientific libraries. |
| | Apply inclusion criterion 1 and exclusion criteria 1-4 | The corresponding inclusion & exclusion criteria are applied on the papers found in the previous process. The full texts of the resulting papers are gathered and organized in reference manager software and imported in data analysis software. |
| | Perform keyword query | The following keyword query is applied using data analysis software: *"pattern OR type OR level OR degree OR approach"*, to simplify the keyword scanning process. |
| MT Levels Identification | Scan for multi-tenancy levels | The full texts are scanned for MULTI-TENANCY LEVELS at the places where keywords are found |
| | Check for citation | When MULTI-TENANCY LEVELS are found, it is checked if the MULTI-TENANCY LEVEL includes a citation to other papers or not. |
| | Scan multi-tenancy level in reference | If so, the reference is scanned to identify the original MULTI-TENANCY LEVEL. |
| | Extract multi-tenancy level | When an original MULTI-TENANCY LEVEL is found, it is extracted from the article using data analysis software. |
| Attributes Identification | Scan for multi-tenancy definition | When no MULTI-TENANCY LEVEL is found, the introduction of the definition of multi-tenancy in the text is searched. If this cannot be identified, the next scientific paper from the list of selected literature is examined. |
| | Scan for criterion | If a MULTI-TENANCY LEVEL or definition is identified, the full text is scanned for consequences, benefits, drawbacks, or requirements. |
| | Extract criterion | When a CRITERION is found, it is extracted from the article using data analysis software. |

Table 5: MT Levels and Criteria Identification Concept Table

| Concept | Definition |
|---|---|
| LIST OF ALL LITERATURE | A list of all the results from applying the search string on the various digital scientific libraries. This concept is subjected to inclusion criterion 1 and exclusion criteria 1-4. |
| LIST OF CANDIDATE LITERATURE | A list containing the scientific articles resulted from using inclusion criterion 1 and exclusion criteria 1-4 on the LIST OF ALL LITERATURE. This concept is organized in reference manager software and imported in data analysis software. It is subjected to a keyword query. |
| LIST OF RELEVANT LITERATURE | A list of scientific articles resulted from using the following keyword query in the data analysis software: *"pattern OR type OR level OR degree OR approach"*. |
| LIST OF SELECTED LITERATURE ON MULTI-TENANCY LEVELS | A list of scientific articles resulted from using inclusion criterion 2 and exclusion criterion 5 on the LIST OF RELEVANT LITERATURE. |
| SELECTED LITERATURE ON MULTI-TENANCY LEVELS | A single article from the LIST OF SELECTED LITERATURE ON MULTI-TENANCY LEVELS. |
| MULTI-TENANCY LEVEL | A certain level, layer, or tier on which multi-tenancy can be applied, or a degree or approach which involves multi-tenancy. |
| LIST OF SELECTED LITERATURE ON CRITERIA | A list of scientific articles resulted from using exclusion criterion 6 on the LIST OF RELEVANT LITERATURE. |
| SELECTED LITERATURE ON CRITERIA | A single article from the LIST OF SELECTED LITERATURE ON CRITERIA. |
| CRITERION | A benefit, drawback, consequence, requirement or consideration related to multi-tenancy. |

scope is maximized. If decision makers nevertheless feel the set of attributes covered by the Multi-Tenant Architecture Selection Model insufficiently represents all of their interests, extra criteria can be manually added to the model.

The first process of the third phase, *Scan for multi-tenancy definition*, is only carried out when no multi-tenancy levels are found. In that case, the text of the article is scanned for a description of the term multi-tenancy. When no description is found, the next article from the LIST OF RELEVANT LITERATURE is scanned for multi-tenancy levels. The second process, *Scan for criterion*, can be accessed in two ways. It is responsible for scanning criteria at the description of the term multi-tenancy or at previously identified multi-tenancy levels. When a criterion is found, the corresponding article is included in the LIST OF SELECTED LITERATURE ON CRITERIA.

Study selection results of the phases will be tabulated as follows:

- number of all literature studies per library source;
- number of relevant literature studies library per source;
- number of candidate literature studies library per source;

## 4.4 Data Extraction Strategy

Data extracted consists of the multi-tenancy levels from the articles in the LIST OF SELECTED LITERATURE ON MULTI-TENANCY LEVELS in the second phase and the criteria from the studies in the LIST OF SELECTED LITERATURE ON CRITERIA in the third phase.

## 4.5 Data Analysis Strategy

The *Multi-Tenancy Levels Identification* phase results in a list of original sources. These will be tabulated to show which literature study cited which original sources. The number of studies citing these original sources are counted. Furthermore, the multi-tenancy levels will be tabulated to show from which article – ordered alphabetically by first author name – which multi-tenancy

levels are identified. The number of studies that describe identical multi-tenancy levels will be counted. These findings will be used to construct a series of multi-tenant architectures, answering research SQ. 1.

The results of the *Decision Criteria Identification* phase will also be tabulated to show which literature study described which decision criteria. For each decision criterion the number of studies that describe it will be counted. These results will be used to define a complete, non-redundant list of decision criteria, in which each criterion is measurable and operational, resulting in an answer to the research SQ. 2.

# 5 Findings of Literature Study

This section describes the execution of the literature study, the data extraction and the analysis of the results.

## 5.1 Execution of the Literature Study

The search strings used in each library are shown in Table 6. Each search string looks distinct, because each digital library uses specific practices for search commands. They do have the same effect. Solely the search function of SpringerLink does not allow to search for authors' defined keywords in articles.

Table 6: Search String per Source

| Digital Library | Search String |
|---|---|
| ACM Portal | Keywords:(tenan* or multitenan*) and Abstract:((tenan* or multitenan*) and (saas or software or service or application)) |
| IEEE Xplore | ("Index Terms":tenan* or "Index Terms":multitenan*) and (("Abstract":tenan* or "Abstract":multitenan*) and ("Abstract":software or "Abstract":service or "Abstract":application or "Abstract":saas)) |
| ScienceDirect | Keywords(tenan* or multitenan*) and Abstract((tenan* or multitenan*) and ("software" or "service" or "application" or "saas")) |
| SpringerLink | ab:((tenant or tenancy or multitenancy or multitenant) and (software or service or application or saas)) |
| Scopus | KEY(tenan* or multitenan*) and ABS((tenan* or multitenan) and ("software" or "service" or "application" or "saas")) |

Applying these search strings in the corresponding digital libraries resulted in a total of 534 literature studies. The number of results per library is tabulated in Table 7. Because SciVerse Scopus' content covers journals provided by publishers like IEEE and Springer, there exist many similar results.

Table 7: list of all literature per source

| Digital Library | Date | Results |
|---|---|---|
| IEEE Xplore | 4-Sep-2012 | 153 |
| ACM Portal | 4-Sep-2012 | 20 |
| SpringerLink | 10-Sep-2012 | 128 |
| Scopus | 11-Sep-2012 | 221 |
| ScienceDirect | 10-Sep-2012 | 12 |

After applying the first inclusion criterion and the first to fourth exclusion criteria a total of 109 studies remained. Table 8 shows the results of these selection criteria tabulated as the number of articles per library. They are displayed in numerical order.

Table 8: LIST OF CANDIDATE LITERATURE per source

| Digital Library | Results |
|---|---|
| IEEE Xplore | 68 |
| ACM Portal | 16 |
| SpringerLink | 13 |
| Scopus | 10 |
| ScienceDirect | 2 |

Then the LIST OF CANDIDATE LITERATURE is subjected to a keyword search query. Studies get excluded when none of the keywords is found, resulting in the LIST OF RELEVANT LITERATURE. This list is tabulated per library source in Table 9. The full list can be found in Table 20 in Appendix A.

Table 9: LIST OF RELEVANT LITERATURE per source

| Digital Library | Results |
|---|---|
| IEEE Xplore | 65 |
| ACM Portal | 17 |
| SpringerLink | 13 |
| Scopus | 10 |
| ScienceDirect | 1 |

In the second phase in Figure 6 original sources of multi-tenancy levels are identified. These sources are shown in Table 10, the second column represents the frequency $f$ the article is cited by other literature. In the texts of Kwok, Nguyen, and Lam (2008) and Kwok and Mohindra (2008) descriptions of multi-tenancy levels with citations are found. However, upon checking the full texts of the references of these citations, the mentioned multi-tenancy levels could not be identified. Both articles are nevertheless included in Table 10 for reference. Table 21 in Appendix B lists the articles that cite the original sources of the multi-tenancy levels..

Table 10: Referenced Articles

| Article | $f$ |
|---|---|
| Guo et al. (2007) | 8 |
| Chong, Carraro, and Wolter (2006) | 7 |
| Jacobs and Aulbach (2007) | 4 |
| Kwok, Nguyen, and Lam (2008) | 4 |
| Z. Wang et al. (2008) | 4 |
| Osipov et al. (2009) | 2 |
| Aulbach, Grust, Jacobs, Kemper, and Rittinger (2008) | 1 |
| Kwok and Mohindra (2008) | 1 |
| Natis and Knipp (2008) | 1 |
| Reinwald (2010) | 1 |
| Taylor and Guo (2007) | 1 |
| Waidner (2009) | 1 |

Extra articles not included in the LIST OF RELEVANT LITERATURE but added to the LIST OF

SELECTED LITERATURE ON MULTI-TENANCY LEVELS because they are cited by other articles, are from Chong et al. (2006); Natis and Knipp (2008); Osipov et al. (2009); Reinwald (2010); Taylor and Guo (2007); Waidner (2009).

## 5.2 Data Extraction

Data extracted from each study consists of the identified multi-tenancy levels and decision criteria as described in Section 4.4. According to Watson and Webster (2002), a literature review should be concept-centric in contrast to author-centric. These concepts correspond to the multi-tenancy levels and criteria in this work. As recommended by Watson and Webster, a concept matrix is constructed showing the concepts discussed in each article.

### 5.2.1 Multi-Tenancy Levels

The concept matrix showing the various multi-tenancy levels is shown in Table 22 in Appendix B. The articles listed in that table form the LIST OF SELECTED LITERATURE ON MULTI-TENANCY LEVELS. Some articles describe the multi-tenancy level *separated*, *dedicated*, *different* or *isolated database* and no description was found on what level resources *are* shared in the data tier (Aghera, Chaudhary, & Kumar, 2012; Domingo et al., 2010; Hui, Jiang, Li, & Zhou, 2009; Kwok, Nguyen, & Lam, 2008; Pippal, Sharma, Mishra, & Kushwaha, 2011; Taylor & Guo, 2007). The shared level of the database server is chosen in these cases. Table 11 lists the extracted multi-tenancy levels and the number of times ($f$) they were identified from these articles.

Table 11: Multi-Tenancy Levels Identified in Selected Literature

| Multi-Tenancy Level | $f$ |
|---|---|
| Application Instance | 16 |
| Database Server | 16 |
| Database | 15 |
| Operating System | 15 |
| Hardware | 14 |
| Schema | 14 |
| Middleware | 12 |
| Virtual Machine | 9 |
| Application Server | 4 |

### 5.2.2 Criteria

Table 23 in Appendix B illustrates from which articles which decision criteria are identified. Table 24 in Appendix B lists all attributes identified from all the articles of the list of selected literature. The number after the attributes defines how many times the corresponding attribute is identified.

## 5.3 Data Analysis

### 5.3.1 Multi-Tenancy Levels

The levels identified from literature at which multi-tenancy can be applied are shown in Table 11. They describe certain levels at which that particular resource can be shared among tenants. These levels can be depicted as layers in a stack with decreasing granularity from top to bottom. Figure 7 illustrates this. A distinction between the application layer and the data layer is made. They correspond to primary layers commonly used in enterprise architecture in order to separate concerns (Fowler, 2002). These different layers don't have to run on different machines, but when a separation is physical, the term tier is often used (Fowler, 2002).

The granularity aspect translates to a sharing versus isolation continuum, where the lowest layer has the lowest level of sharing with the highest level of isolation. For the highest layer it is vice versa. When multi-tenancy is applied at a certain level, the levels below that level are shared among tenants as well, but isolation occurs at the levels above, i.e. for each tenant a

dedicated instance is running. This applies to the application and data layer independently. For example, when multi-tenancy is applied at the application server level, the application server, virtual machine instance and hardware are shared among tenants. Isolation occurs at the levels above the application server, so each tenants receives a dedicated application instance, but multi-tenancy in the data layer can be applied differently.

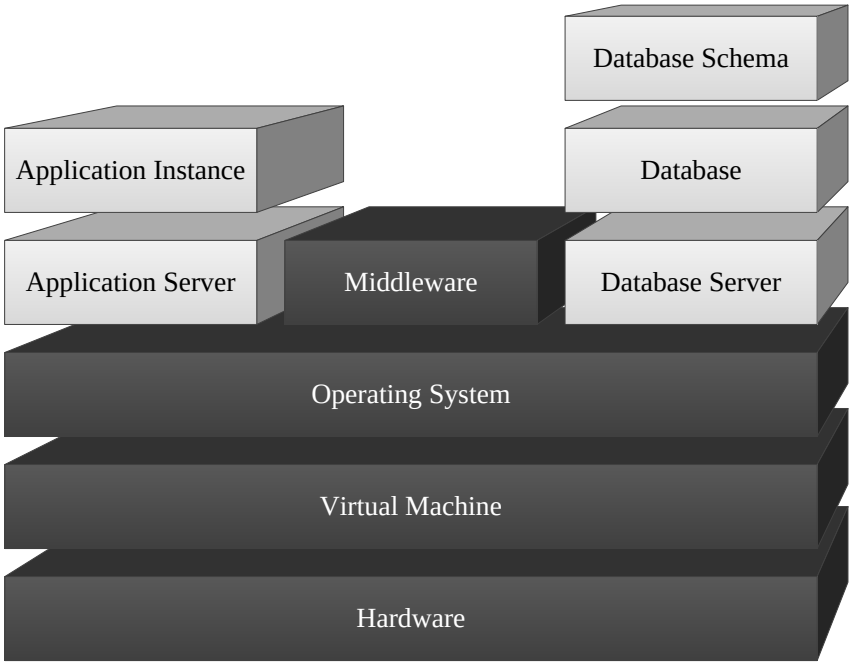



Figure 7: Computing Stack

**Hardware & Virtual Machine** The lowest level on which multi-tenancy can be applied is hardware. It is comprised of processors, storage, memory, networks and other fundamental computing resources. When multi-tenancy is applied at this level, different tenants use the same machines on which their applications are running, but for each tenant a separate virtual machine is running. At the virtual machine level and the levels above, tenants are isolated from one another. Because this multi-tenancy level occurs in the hypervisor layer through virtualization, it is by some called hypervisor-level multi-tenancy (Kurmus, Gupta, Pletka, Cachin, & Haas, 2011) or virtualization-level multi-tenancy (Truyen et al., 2012; Walraven, Truyen, & Joosen, 2011). An example of a service using this type of multi-tenancy is the Amazon Relational Data Service (Azeez et al., 2010). Virtual machines are not required in a computing stack, but are usually used for abstraction of the underlying hardware. When multi-tenancy is applied at the virtual machine level and virtualization is used, tenants are consolidated on the same virtual machine (Truyen et al., 2012). Multi-tenancy at the hardware or virtual machine level is typically applied by Infrastructure as a Service vendors (Mell & Grance, 2011).

**Operating System & Middleware** One level above virtual machine is the operating system. When multi-tenancy is applied at this level, tenants share the same operating system. At this level, the degree of isolation is still relatively high, because for each tenant a dedicated application and database server is running. An example is a Java Virtual Machine running on a operating system process (Rodero-Merino, Vaquero, Caron, Muresan, & Desprez, 2012). The middleware can be depicted at the same level of the application and database servers. It provides services to software applications not available from the operating system. Platform as a service providers are concerned with these levels.

**Application & Database Server** The next level is that of the application and database servers. They refer to computer programs providing services to software applications or other computer programs and do not refer to physical computer hardware systems. When multi-tenancy is applied up to and including this level, the tenants are consolidated on a single server, but for each tenant an isolated instance is running.

**Application Instance** The top of the stack in the application layer refers to the ability to apply multi-tenancy to an application instance. If this is the case, the application is developed in such a way that a single application instance can be offered to multiple tenants concurrently (Guo et al., 2007).

**Database & Schema** The final two levels of the stack in the data layer are database and database schema. A database schema can be regarded as a set of database tables. These two approaches were first described by Chong et al. (2006). When tenants are consolidated in a single database, each tenant operates its own set of tables. In schema-level multi-tenancy, isolation occurs at table row level.

**Number of MTA's** Various architectures comprising of an application and data layer can be constructed from the different multi-tenancy levels depicted in Figure 7. The number of different architectures is the product of the number of options in the application layer and the number of options in the data layer. There are six options in the application layer: the five options from the stack plus no multi-tenancy at all. The same applies to the data layer which sums up to seven options. Therefore, the total amount of different architectures that can be composed is 42.

**Relevant MTA's** In cloud computing, an infrastructure provider manages and controls the infrastructure consisting of processing, storage, networks and other fundamental computing resources (Mell & Grance, 2011). If and to what extent to apply multi-tenancy in the hardware, virtual machine and operating system levels focuses on a load problem. For a service provider, which develops the application and is the primary stakeholder in this research, the aspect of multi-tenancy in the hardware, virtual machine and operating system levels is of significant less importance. It has no influence on the development of the service. The amount of servers, instances and databases is far more relevant for a SaaS provider. For this reason, the three lowest levels are not considered in structuring different types of multi-tenant architectures.

Consequently, a SaaS provider has the following three options in the application layer:

1. A dedicated application server is running for each tenant, and therefore each tenant receives a dedicated application instance.
2. A single application server is running for multiple tenants and each tenant receives a dedicated application instance.
3. A single application server is running for multiple tenants and a single application instance is running for multiple tenants.

The first option corresponds to multi-tenancy enabled at the hardware, virtual machine or operating system level. The second alternative is equal to application server multi-tenancy. The third choice corresponds to multi-tenancy enabled at the application instance level. In the data layer, a service provider can select one the following four options:

1. A dedicated database server is running for each tenant, and therefore each tenant receives a dedicated database.
2. A single database server is running for multiple tenants and each tenant receives a dedicated database.
3. A single database server is running for multiple tenants, data from multiple tenants is stored in a single database, but each tenant receives a dedicated set of tables.

4. A single database server is running for multiple tenants, data from multiple tenants is stored in a single database and a single set of tables.

The first option is equal to multi-tenancy applied at the hardware, virtual machine or operating system level. The second one corresponds to database server multi-tenancy. The third alternative is multi-tenancy to the database and the final one is equal to database schema multi-tenancy.

### 5.3.2 Multi-Tenant Architectures Structuring

From these options in both the application and data layer a set of multi-tenant architectures (MTA's) can be constructed. The number of possible architectures is twelve. They can be displayed in a diagram with two axes, one axis describing the extent of multi-tenancy in the application layer, the other axis in the data layer. This diagram is shown in Figure 8.

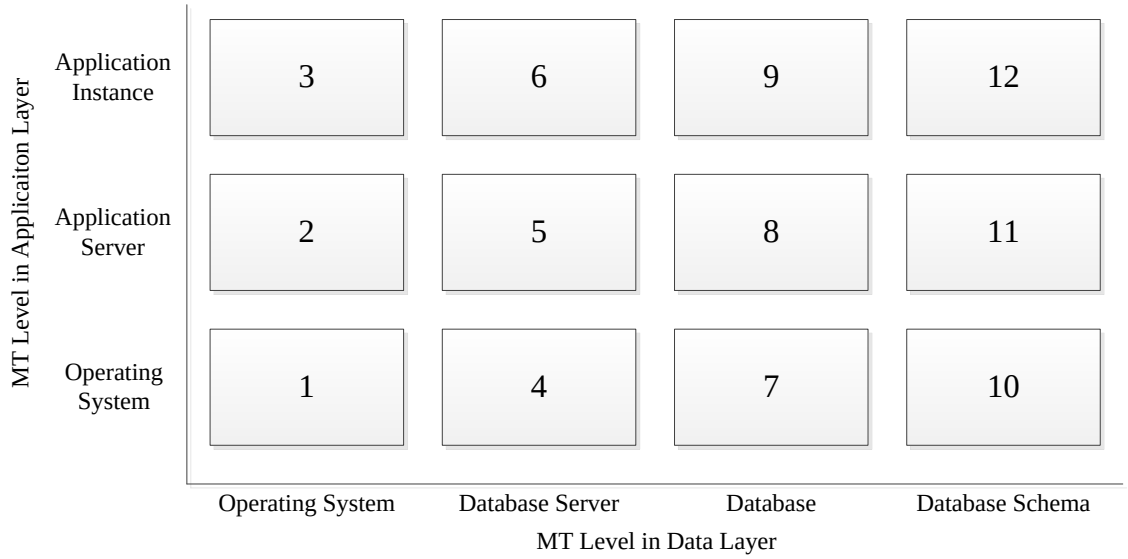



Figure 8: Architecture Diagram

Each individual architecture is schematically displayed in Figures 10 to 21 as a model in which three tenants (Tenant A, B and C) communicate with a software application consisting of an application layer and a data layer. The MTA's are numbered and correspond to the numbering in Figure 8. The application layer is represented as a set of application servers running one or multiple application instances. The data layer is displayed as a set of database servers, running one or more databases, in which one or multiple database schema's exist. If one of these entities is shared among the tenants, its color is gray. If its dedicated to only one tenant, its colored white. For the sake of simplicity only three tenants are displayed in the architectures. A SaaS provider can of course offer his software application to more than three tenants, the models merely presents possible arrangements of shared resources. The symbols used in the representation of the multi-tenant architectures are shown in Figure 9.

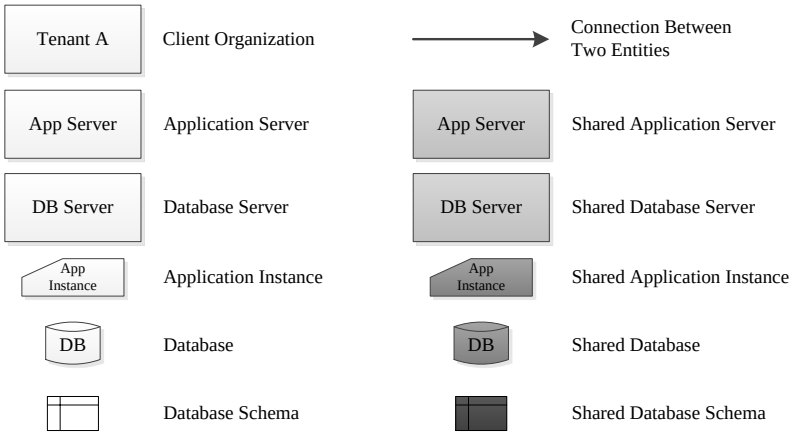



Figure 9: Symbols

**MTA 1** The first option describes an architecture in which the tenants share no resources at all. For each tenant runs a dedicated application server (AS) and a dedicated database server (DBS). This architecture is graphically explained in Figure 10.

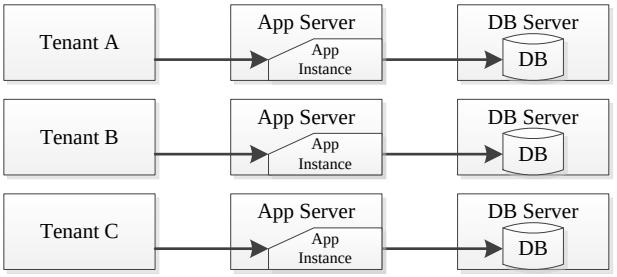



Figure 10: MTA 1 with a Dedicated AS & a Dedicated DBS

**MTA 2** The second alternative describes an architecture in which the tenants only share a common application server. In this server a dedicated application instance is running for each tenant. They receive a dedicated database server as well. This architecture is illustrated in Figure 11.

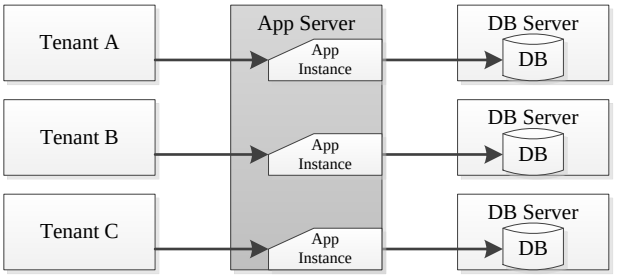



Figure 11: MTA 2 with a Shared AS & a Dedicated DBS

**MTA 3** The final option in which each tenant receives a dedicated database server is an architecture in which the tenants share the application instance in addition to the application server. The shared application instance communicates to dedicated databases for each corresponding tenant. This is shown in Figure 12.

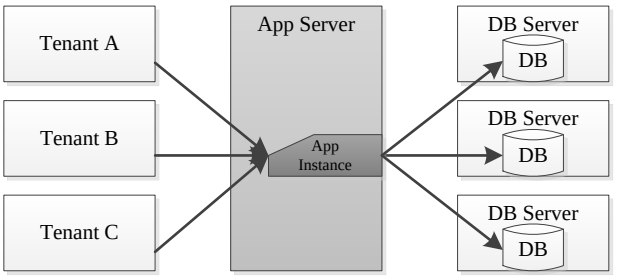



Figure 12: MTA 3 with a Shared Application Instance & a Dedicated DBS

**MTA 4** The next three multi-tenant architectures cover those architectures in which the tenants share a database server. The database server still contains a dedicated database for each tenant. In the first architecture of that series, a dedicated application server is running for each tenant, see Figure 13.

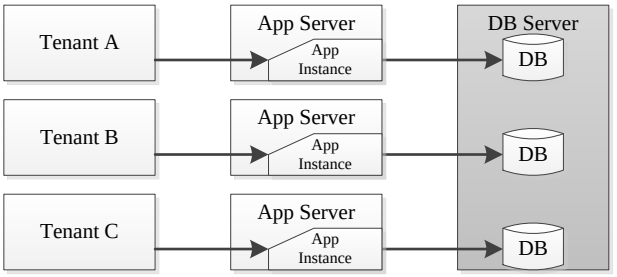



Figure 13: MTA 4 with a Dedicated AS & a Shared DBS

**MTA 5** Another solution in that series is to make the application server multi-tenant. In this case, tenants share an application server, in which for each tenant an isolated application instance is running. These instances communicate with dedicated databases. This architecture is illustrated in Figure 14.

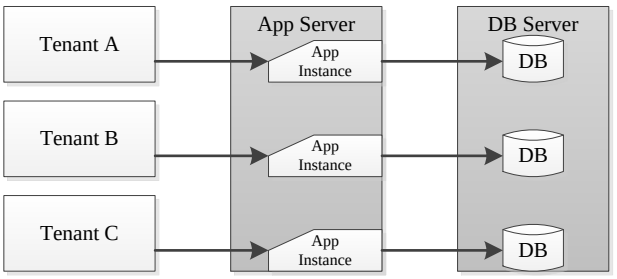



Figure 14: MTA 5 with a Shared AS & a Shared DBS

**MTA 6** The final alternative architecture in which a shared database server exists, combines this with a shared application instance. This instance is running on a shared application server and communicates with dedicated databases. The corresponding figure is presented in Figure 15.

**MTA 7** The next three multi-tenant architectures have a shared database in a shared database server in common. Isolation in the data tier occurs at the database table level, i.e. each tenant communicates with a dedicated set of database tables. The first of these three architectures is the one in which each tenant receives a dedicated application server. It is graphically described in Figure 16.

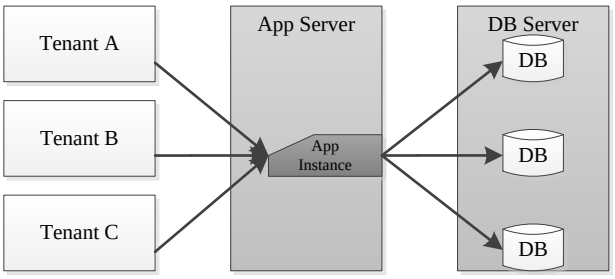



Figure 15: MTA 6 with a Shared Application Instance & a Shared DBS

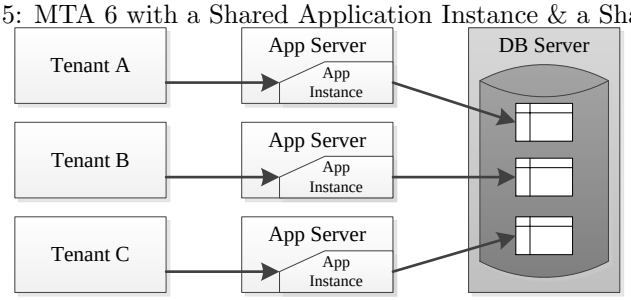



Figure 16: MTA 7 with a Dedicated AS & a Shared DB

**MTA 8** The next architecture combines the shared database with a shared application server. For each tenant a dedicated application instance is running in the application server communicating with a dedicated database table. The architecture is illustrated in Figure 17.

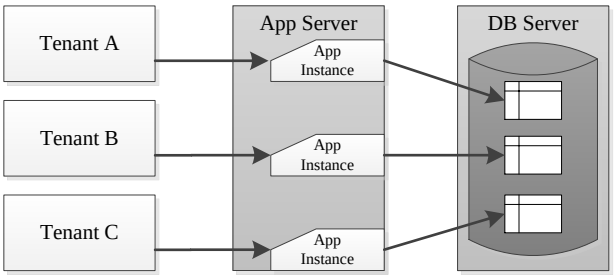



Figure 17: MTA 8 with a Shared AS & a Shared DB

**MTA 9** The final architecture in the shared database series is the one in which tenants share a application instance. It is running on a shared application server and communicates with dedicated sets of database tables. The corresponding figure is shown in Figure 18.

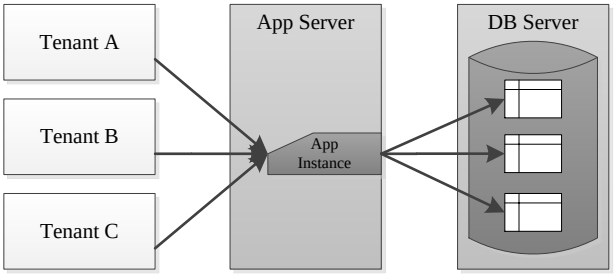



Figure 18: MTA 9 with a Shared Application Instance & a Shared DB

**MTA 10** In the last three multi-tenant architectures tenants share a set of database tables. Isolation the data tier occurs at database table row level. The first of these three architectures combines an application tier in which for each tenant a dedicated application server is running. The architecture is presented in Figure 19.

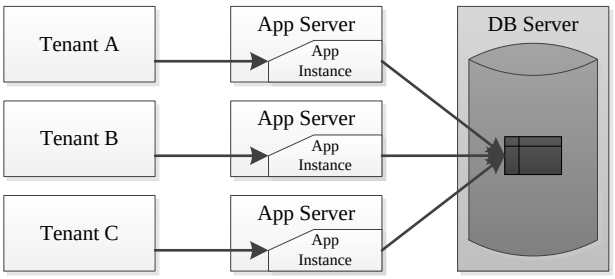



Figure 19: MTA 10 with a Dedicated AS & a Shared DB schema

**MTA 11** The next architecture combines the shared database table with an application tier in which tenants share an application server. They receive a dedicated application instance, see Figure 20.

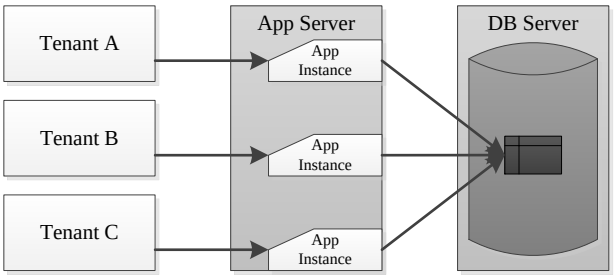



Figure 20: MTA 11 with a Shared AS & a Shared DB schema

**MTA 12** The very last multi-tenant architecture is illustrated in Figure 21. It shows it combines a shared set of database tables with a shared application instance.

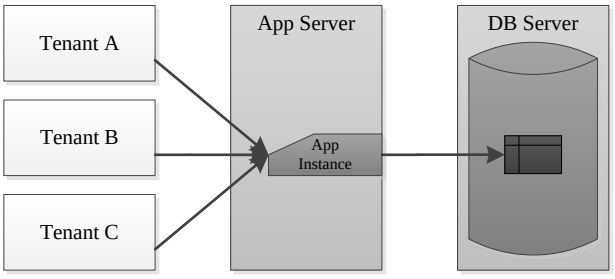



Figure 21: MTA 12 with a Shared Application Instance & a Shared DB Schema

This concludes the structuring of the multi-tenant architectures and thereby the analysis of the multi-tenancy levels. The next section describes the analysis of the decision criteria.

### 5.3.3 Criteria

**Minimizing the Criteria Set** A total of 106 criteria are initially identified from literature, see Table 24. This set still includes irrelevant and redundant attributes. Thus, it does not yet adhere to the *minimal* and *non-redundant* principle, mentioned in Section 3.2.2. The list of criteria should

therefore be reduced. Figure 22 illustrates the process how the initial list of criteria is minimized. The activity and concept table belonging to the PDD in Figure 22 are represented as Tables 12 and 13, respectively.

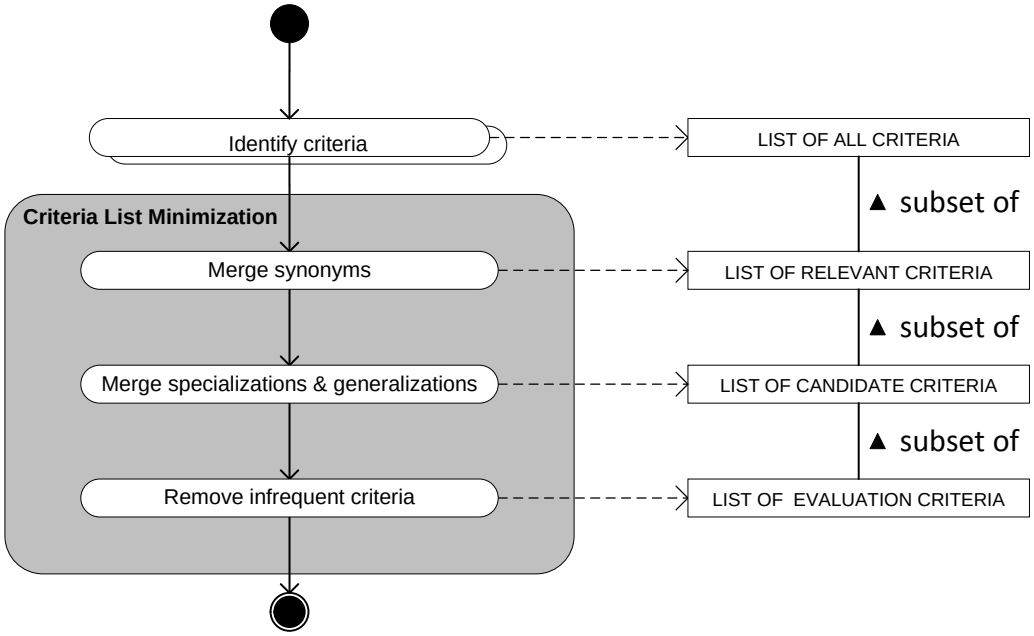



Figure 22: PDD to Minimize the Criteria List

Table 12: Reducing Criteria List Activity Table

| Sub-Activity | Description |
|---|---|
| Identify criteria | This activity is comprised of the sub-activities in the *Criteria Identification* activity in Figure 6 |
| Merge synonyms | Criteria that are synonyms, or have the same meaning, or define equalities are merged. |
| Merge specializations & generalizations | Criteria that define specializations of generalizations are merged. |
| Remove infrequent criteria | Criteria that are identified less than five times are deleted from the final criteria list. |

**Minimization Steps** The first step in reducing the initial large set of decision criteria is by merging synonyms. Those criteria that have an equal meaning an define equalities are merged in this step too. Then, criteria that represent specializations of other criteria are combined. This only happens when the specializations define concepts so narrow, that assessing the multi-tenant architectures on these criteria is considered ineffective and inefficient. The final step to reduce the list is by deleting infrequent criteria, i.e. attributes that are identified less than five times in literature. How the criteria list is subjected to these steps is shown in Figures 27 to 29 in Appendix C. The majority of the combinations will be straightforward, but on three notable

Table 13: Reducing Criteria List Concept Table

| Concept | Definition |
|---|---|
| LIST OF ALL CRITERIA | A list of all the criteria identified from the LIST OF SELECTED LITERATURE. The LIST OF ALL CRITERIA is displayed in Table 24 in Appendix B. |
| LIST OF CANDIDATE CRITERIA | A list of the criteria where synonyms in LIST OF ALL CRITERIA are merged. |
| LIST OF SELECTED CRITERIA | A list containing criteria where specializations of generalizations from LIST OF CANDIDATE CRITERIA are merged. |
| LIST OF EVALUATION CRITERIA | A list of criteria that omits the criteria mentioned less than five times from LIST OF SELECTED CRITERIA. |

decisions is expanded.

**Scalability** Scalability is quite often identified in literature as a concept of interest when multi-tenancy is mentioned or described. Bondi (2000) defines scalability as a desirable ability of a system, network, or process to accommodate an increasing amount of elements and process this accompanying extra volume of work in a capable manner. Additional workload is required when the service is offered to extra tenants or users. As a result, scalability is related to both the number of tenants and the number of end-users an architecture can support. For that reason, scalability is merged with both those corresponding attributes.

**Fault Tolerance** In the elaborated Computer Science Handbook (Tucker, 2004), fault tolerance and the metrics reliability and availability are defined. Fault tolerance is described as "the total number of failed elements that can be present without causing output errors" (Tucker, 2004, p. 649). The reliability of a system is defined as "...the probability that the system will produce correct outputs up to time $t$, provided it was producing correct outputs to start with" (Tucker, 2004, p. 646). A highly reliable system will produce correct outputs for a long time, even when there are failed elements. This requires a high fault tolerance. The availability of a system is defined as "...the probability that the system is operational at time $t$" (Tucker, 2004, p. 646). A highly available system will stay operational even when failed elements occur. Again, this requires a high fault tolerance. Therefore, fault tolerance is merged with both reliability and availability.

**Access Control** Access control and its relationships with authorization and authentication is extensively discussed by Sandhu and Samarati (1994). The authors define access control as "to limit the actions or operations that a legitimate user of a computer system can perform. Access control constrains what a user can do directly, as well what programs executing on behalf of the users are allowed to do. In this way access control seeks to prevent activity which could lead to a breach of security" (1994, p. 1). Authentication is concerned with "correctly establishing the identity of the user" (Sandhu & Samarati, 1994, p. 1). Authorization occurs "to determine if the user attempting to do an operation is actually authorized to perform that operation" Sandhu and Samarati (1994, p. 1). The effectiveness of access control depends on proper authentication. and correct authorization (Sandhu & Samarati, 1994). Therefore, access control is merged with both corresponding criteria.

**Criteria After Minimization** The result of the activities listed in Table 12 is the LIST OF EVALUATION CRITERIA, which is illustrated in Table 14. The value after each decision criterion shows how many times ($f$) that criterion is identified from the LIST OF SELECTED LITERATURE. This list does not yet adhere to the *operational* principle described in Section 3.2.2. To obtain

this property, the list needs to be evaluated by experts – which is why this list is given the name: list of evaluation criteria. More on this evaluation process is discussed in Section 6.

Table 14: Evaluation Criteria

| Criterion | $f$ | Criterion | $f$ |
|---|---|---|---|
| Variability | 65 | Authorization | 11 |
| Number of Tenants | 60 | Response Time | 11 |
| Security | 48 | Operating Cost | 10 |
| Maintenance | 45 | Deployment Time | 9 |
| Number of End-Users | 44 | Flexibility | 9 |
| Resource Utilization Efficiency | 42 | Throughput | 8 |
| Performance | 32 | Monitoring | 7 |
| Software Complexity | 32 | Diverse SLA | 5 |
| Recoverability | 23 | Migration | 5 |
| Availability | 16 | Reliability | 5 |
| Authentication | 12 | | |

The criterion *operating cost* covers a broad range of expenses, e.g. business overhead costs and equipment operating costs. All attributes in Table 14 can be associated with certain types of costs. The criterion *operating cost* encompasses most costs associated with these other attributes. For this reason, *operating cost* will not be included as a decision criterion in the Multi-Tenant Architecture Selection Model.

**Matching with Quality Characteristics** Some of the criteria from Table 14 are equal or synonymous to the quality characteristics of software products and computer systems defined in ISO/IEC 25010 (ISO, 2011). These quality characteristics are used to define the quality of software and computer systems and are part of a so-called product quality model. In addition, "..., many of the characteristics are also relevant to wider systems and services" (ISO, 2011, p. 1). So the characteristics are applicable on multi-tenant architectures too. The links made between the criteria and the quality characteristics are shown in Table 15. The number after each quality characteristic represents the numbering of the section in ISO/IEC 25010 and can be seen as a subdivision between the characteristics. Most connections are elementary, but two connections are explained in more detail.

**Capacity** The criteria throughput, number of tenants, and number of end-users can be grouped under the characteristic capacity. The description of capacity in ISO/IEC 25010 states that parameters influencing the capacity are, inter alia, the throughput and the number of concurrent users. In reviewing the capacity of a multi-tenant system, the number of tenants is a parameter of the capacity as well. Because these three criteria can be categorized as capacity, it is added to the list of decision criteria. However, the three individual criteria are considered to be too important and distinct to be excluded and are therefore still part of the list of decision criteria.

**Authorization** The authorization criterion is linked with two different quality characteristics. Confidentiality is the term used to ensure "...that information is not made available or disclosed to unauthorized individuals, entities or processes" (ISO, 2012, p. 2). Integrity means that information can not be modified undetectably. Both characteristics cover the aspect of authorization, therefore authorization itself is omitted from the list and confidentiality and integrity are used instead.

**Unmatched Criteria** The quality characteristic adaptability initially looks like a match for flexibility. However, adaptability as described in ISO/IEC 25010 is concerned with the degree to

Table 15: Evaluation Criteria linked to Quality Characteristics

| Evaluation Criterion | | Quality Characteristic | ISO Hierarchy |
|---|---|---|---|
| Performance | → | Performance Efficiency | ..2 |
| Response Time | → | Time Behavior | ..2.1 |
| Resource Utilization Efficiency | → | Resource Utilization | ..2.2 |
| Throughput<br>Number of Tenants<br>Number of End-Users | ↘<br>↗ | Capacity | ..2.3 |
| Reliability | → | Reliability | ..5 |
| Availability | → | Availability | ..5.2 |
| Recoverability | → | Recoverability | ..5.4 |
| Security | → | Security | ..6 |
| Authorization | ◁ | Confidentiality<br>Integrity | ..6.1<br>..6.2 |
| Authentication | → | Authenticity | ..6.5 |
| Maintenance | → | Maintainability | ..7 |
| Migration | → | Portability | ..8 |
| Deployment Time | | - | |
| Flexibility | | - | |
| Variability | | - | |
| Diverse SLA | | - | |
| Software Complexity | | - | |
| Monitoring | | - | |

which a product or system can be adapted and flexibility as the evaluation criterion is concerned with the degree to which a product or system itself can support various usage environments. The evaluation criterion deployment time is not matched with the quality characteristic installability, because installation is considered a sub-activity of software deployment. As such, these two decision criteria are not matched with quality characteristics. For the other four decision criteria no corresponding quality characteristic has been found. This is not considered as a shortcoming of ISO/IEC 25010. There are additional factors influencing the use of multi-tenancy than just characteristics measuring the quality of software products and computer systems. Multi-tenant architectures are a special type of systems and the product quality model of ISO/IEC 25010 cannot account for all these specific systems. For those criteria that are linked to a comparable quality characteristic, their name is changed to that of the quality characteristic. Also, the descriptions of the quality characteristics are used to describe the decision criteria.

**Decomposable Property** The product quality model decomposes the quality properties in characteristics and sub-characteristics. This subdivision is shown by the numbering of the quality characteristics, see the fourth column in Table 15. The same composition of characteristics and sub-characteristics is applicable to the decision criteria. This supports the *decomposition* of decision criteria, another principle of the set of attributes, see Section 3.2.2.

**Resulting Criteria** A clear description of the decision criteria is essential for a thorough understanding of them. When using the Multi-Tenant Architecture Selection Model, decision makers will need to weigh these criteria to calculate the relative importance and experts need to judge the performance of the architectures with respect to these criteria. For that reason, the descriptions of the quality characteristics are used to describe the criteria. This applies only for those criteria that are matched with a quality characteristic. For the other six criteria descriptions are created. Table 16 shows the list of the decision criteria together with its description. These criteria are selected for evaluation.

Table 16: Criteria Set

| | Criterion | Description |
|---|---|---|
| 1 | Maintainability | Degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers. Modifications can include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specifications. Modifications include those carried out by specialized support staff, and carried out by business or operational staff, or end users. It includes installation of updates and upgrades. |
| 2 | Security | Degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization. Survivability (the degree to which a product or system continues to fulfill its mission by providing essential services in a timely manner in spite of the presence of attacks) is covered by recoverability. |
| 2.1 | Confidentiality | Degree to which a product or system ensures that data are accessible only to those authorized to have access. |
| 2.2 | Integrity | Degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data. |
| 2.3 | Authenticity | Degree to which the identity of a subject or resource can be proved to be the one claimed. |
| 3 | Portability | Degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another. |
| 4 | Performance | Performance relative to the amount of resources used under stated conditions. |
| 4.1 | Time Behavior | Degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements. |
| 4.2 | Resource Utilization | Degree to which the amounts and types of resources used by a product or system when performing its functions meet requirements. |
| 4.3 | Capacity | Degree to which the maximum limits of a product or system parameter meet requirements. Parameters can include the number of items that can be stored, the number of concurrent users, the communication bandwidth, throughput of transactions, and size of database. |
| 4.3.1 | Throughput | The average rate of successful message delivery over a communication channel, measured in bits per second. |
| 4.3.2 | Number of Tenants | The extent to which the system can be scaled so it can be offered to multiple tenants. |
| 4.3.3 | Number of End-Users | The extent to which the system can be scaled so it can be offered to multiple end-users. |
| 5 | Flexibility | Degree to which the system can support different functional and non-functional requirements of different tenants. |
| 5.1 | Variability | Degree to which the system can support customized solutions and tenant-dependent configurations, extension and evolution. |
| 5.2 | Diverse SLA | Degree to which the system can support a variety of service level agreements to tenants. |
| 6 | Software Complexity | Degree of the software complexity of the software product if it is developed and implemented in a multi-tenant architecture. |
| 7 | Reliability | Degree to which a system, product or component performs specified functions under specified conditions for a specified period of time. |
| 7.1 | Availability | Degree to which a system, product or component is operational and accessible when required for use. |
| 7.2 | Recoverability | Degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system. |
| 8 | Monitoring | Degree of ease to which monitoring and controlling tasks can be carried out in the system. Tasks include controlling server availability, user activity, capacity and performance. |
| 9 | Deployment Time | Degree of effectiveness and efficiency to which the software product can be made available for use. |

**Recap** The structuring of the twelve multi-tenant architectures and the creation of this set of decision criteria completes the first phase of the Research Model depicted in Figure 4. The twelve multi-tenant architectures have been constructed by identifying out literature various levels at which multi-tenancy can be applied. Also, a list of consequences, requirements and pros and cons

related to multi-tenancy has been identified from which a set of decision criteria is created. These two elements are an essential part of the final Multi-Tenant Architecture Selection Model.

# 6 Evaluation of Multi-tenant Architectures and Criteria

Prior to collecting performance values of the multi-tenant architectures on each decision criterion, both these elements should be evaluated with experts. Reason to evaluate the multi-tenant architectures is that the technical feasibility of the twelve architectures in Figures 10 to 21 was excluded in structuring those multi-tenant architectures. Of course, these architectures require to represent a realistic architecture, that may be used in a real system. Therefore, they need to be evaluated on the extent to which they represent a feasible architecture.

As mentioned in Section 3.2.2, the set of decision criteria should meet five principles. One of those is that the set of criteria should be operational. This means that the attributes should represent decision criteria that are important to the decision makers. In addition, the attributes should discriminate among the different multi-tenant architectures, thus it should be expected that the performance values of the architectures evaluated on the criteria differ. The set of decision criteria should be evaluated on both these points.

If the architectures and the criteria don't meet these requirements, they can be omitted from the final Multi-Tenant Architecture Selection Model.

## 6.1 Strategy

The research method used for the evaluation of the multi-tenant architectures and criteria is a survey. This method is chosen, because it consists of systematic and standardized approaches for collection data on individuals (Marsden & Wright, 2010). The collected data on individuals represent the expert opinions on the multi-tenant architectures and decision criteria regarding the evaluation points discussed in the previous section.

### 6.1.1 Instrument

The instrument, i.e. the data collection method, is a questionnaire. Questionnaires are considered less time-consuming than interview surveys. The questionnaires are administered on-site by an evaluator so questions can be asked by and clarifications can be provided to the experts when needed. The questionnaire questions are written out on paper and handed over to the respondent. The native language of both the experts and evaluator is Dutch. To ease communication, the questionnaire is written in Dutch. An English translation of the questionnaire can be found in Appendix D.

### 6.1.2 Experts

Expert selection is based on job function. The following job functions are selected: architect, software development manager or technical manager. They are chosen, because experts with one of these job functions have an in-depth technical understanding on the structure of complete systems and software applications. All respondents work in the same large IT organization on different software applications. A total of ten experts are involved in the evaluation.

### 6.1.3 Format

The on-site administered questionnaire starts with the evaluator explaining the goal of this research, the Multi-Tenant Architecture Selection Model to be developed and the identification process of the twelve multi-tenant architectures. Then, an introductory page and the schematics of the multi-tenant architectures (Figures 10 to 21) are handed over to the respondent. A second piece is handed over containing the actual questions. First, general open-ended questions like name, date, job function, years of service and product type are asked. Thereafter, for each multi-tenant architecture the extent to which it represents a feasible architecture is asked. Subsequently, for each criterion the extent to which it discriminates among the architectures, and the extent to which it defines a deciding factor are asked. These are closed questions with an interval-level response format. Experts are presented with a 7-point Likert scale. Reason this scale is used,

is because it provides a more reliable outcome when compared to lower point scales (Preston & Colman, 2000). In addition, it's still fairly easy to describe the differences between scale attributes semantically as opposed to the 10-point scale.

### 6.1.4 Analysis Procedure

Inclusion of the architectures and criteria in the Multi-Tenant Architecture Selection Model depends on the median of the evaluation scores given by the experts. The median describes a numerical value separating the higher half of a list of numbers from the lower half. If the list has an even number of items, the median is defined as the mean of the two middle values. The median is a more robust measure of central tendency in the presence of outlier values than the mean is. Moreover, the value of the median is more easily translated to a semantic description than that of the mean. All answers in the questionnaire use a 7-point Likert scale, so the lowest possible number is a 1 and the highest possible number is a 7. If an architecture's median is equal to or greater than 3, it's included in the decision support model. This threshold is chosen, because the third Likert item is semantically described as *slightly* feasible and is considered as sufficiently feasible to be included in the decision support model. The decision criteria are evaluated on two requirements, therefore inclusion of a decision criterion depends on two medians. If these medians are both equal to or greater than 3, the corresponding criterion is included in the decision support model. This threshold is chosen, because the third Likert item, described as a *slight* discrimination or deciding factor, is considered as sufficient.

## 6.2 Findings

This section presents the data of the completed evaluation questionnaires. Data is presented in Tables 17 to 18. The columns, excluding the last one, represent the 7-point Likert scale. The elements of these columns define the *frequency* experts answered a question with the corresponding Likert item. The final column defines the median of the frequencies for the corresponding architecture. It is denoted by $\mu_{\frac{1}{2}}$. The frequencies of both the discrimination and deciding factor are combined in a single table.

### 6.2.1 Multi-tenant Architectures Evaluation on Feasibility

Table 17 shows the answers of the experts on the extent of feasibility per architecture. MTA 1 receives a high degree of feasibility, seven experts defined it as at least a very strongly feasible architecture. The opinions are more divided MTA 2, but the majority agrees it represents at least a moderately feasible architecture. MTA 3 proves to be the lowest feasible architecture with an aggregate value between slightly and moderately. Three experts define MTA 4 as slightly feasible, yet five experts define it is at least very strongly feasible. On MTA 5 no real consensus is reached as well, but the collective is stated as having a value between moderately and strongly feasible. Opinions on MTA 6 are even more divided as each Likert item is checked. Its shared value is moderately feasible. The extent of feasibility on MTA 7 is a bit higher with a value between moderately and strongly. Half of the experts define MTA 8 as at least very strongly feasible. On the extent of feasibility on MTA 9, the judgments can divided in two equally large groups. One stating it is slightly feasible at best, and the other stating at least strongly feasible. The joint value however is moderately feasible. Six out of ten experts define MTA 10 as a strongly feasible architecture, equal to its aggregate value. MTA 11 receives a strong degree of feasibility and on MTA 12 experts concur on a very strong degree of feasibility. For each median $\mu_{\frac{1}{2}}$ in Table 17, $\mu_{\frac{1}{2}} \geq 3$ applies and therefore all architectures will be included in the Multi-Tenant Architecture Selection Model. These medians are not calculated to identify differences among the architectures, but to check per architecture individually how they score on the feasibility aspect. Therefore, the amount of difference between the maximum and minimum value is of minor importance.

Table 17: Frequencies of Chosen Likert Items on the Feasibility of Multi-tenant Architectures

| | Multi-Tenant Architecture | | Feasibility 1 | 2 | 3 | 4 | 5 | 6 | 7 | $\mu_{\frac{1}{2}}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Dedicated AS | & Dedicated DBS | 1 | 1 | 0 | 0 | 1 | 2 | 5 | 6.5 |
| 2 | Shared AS | & Dedicated DBS | 0 | 1 | 2 | 3 | 2 | 1 | 1 | 4.0 |
| 3 | Shared Instance | & Dedicated DBS | 1 | 2 | 2 | 4 | 1 | 0 | 0 | 3.5 |
| 4 | Dedicated AS | & Shared DBS | 0 | 0 | 3 | 1 | 1 | 2 | 3 | 5.5 |
| 5 | Shared AS | & Shared DBS | 0 | 0 | 2 | 3 | 1 | 2 | 2 | 4.5 |
| 6 | Shared Instance | & Shared DBS | 1 | 1 | 2 | 2 | 1 | 2 | 1 | 4.0 |
| 7 | Dedicated AS | & Shared DB | 0 | 0 | 2 | 3 | 2 | 1 | 2 | 4.5 |
| 8 | Shared AS | & Shared DB | 0 | 0 | 2 | 2 | 1 | 3 | 2 | 5.5 |
| 9 | Shared Instance | & Shared DB | 1 | 2 | 2 | 0 | 2 | 2 | 1 | 4.0 |
| 10 | Dedicated AS | & Shared Schema | 1 | 0 | 1 | 1 | 6 | 1 | 0 | 5.0 |
| 11 | Shared AS | & Shared Schema | 1 | 0 | 1 | 1 | 3 | 4 | 0 | 5.0 |
| 12 | Shared Instance | & Shared Schema | 1 | 2 | 0 | 0 | 1 | 5 | 1 | 6.0 |

### 6.2.2 Decision Criteria Evaluation on Discrimination & Deciding Factor

During one questionnaire with an expert, he admitted he did not had sufficient knowledge to answer the questions related to the extent of deciding factor. This is reflected in Table 18 as the frequencies on the deciding factor of a decision criterion sum up to nine instead of ten.

Performance Efficiency is seen by the experts as having a high discriminating and deciding factor. This applies to Time Behavior as well. According to the experts, Resource Utilization holds a high discrimination, but on the deciding factor opinions are more divided. Still, more than half define this criterion as having at least a strong deciding factor. Experts agree that Capacity has at least a strong discrimination and a strong deciding factor. On Throughput, they are more divided. Four experts state Throughput as having a slight discrimination, but five experts state is has at least a strong discrimination. The aggregate value equates to moderate. On the deciding factor of Throughput, no Likert item is checked more than twice. Here too, the aggregate value equals moderate. There is a better consensus on the Number of Tenants criterion with a high discrimination factor and a very strong deciding factor. Slightly less agreement exists for the discrimination on the Number of End-users criterion, but still seven experts define it at least as strong discriminating. Again a high consensus on the deciding factor with a value equal to very strong. For Reliability there are six experts stating it has at least a strong discrimination and seven experts stating it has at least a strong deciding factor. This is also the case for the Availability criterion. For Recoverability, the number of experts sharing the opinion that it has at least a strong discrimination and deciding factor is even higher. On Security and Confidentiality, all experts share that opinion. There is less consensus on both factors of the Integrity criterion, yet six experts find it has at least a strong discrimination and deciding factor. Authenticity receives the lowest aggregate value on discrimination. Six experts state it discriminates slightly among the architectures at best and five experts state it has at least a strong deciding factor. On the Maintainability criterion experts are in a high agreement stating it has both a very strong discrimination and is a very strong deciding factor. On Portability too, there is a good consensus, but the degree of discrimination and deciding factor is only moderate. The aggregate values for Deployment Time is between those of Maintainability and Portability. This applies to Flexibility as well. Eight experts state Variability discriminates to a degree between moderately and very strong. Seven experts state its deciding factor lies on moderate or strong. For Diverse SLA there are again eight experts stating it discriminates moderately to very strong. A majority answered the deciding factor with a moderate extent. For Software Complexity there are seven experts defining it as at least discriminating strongly and six experts defining it as an at least strong deciding factor. Finally, there exists a high consensus for Monitoring where nine experts agree it discriminates strong or very strong and seven experts agree it has a strong or very strong deciding

factor. All medians for each decision criterion are equal to greater than 3 and therefore each decision criterion is included in the final decision model.

## 6.3 Accommodation of the Multi-Tenant Architecture Selection Model

Now the evaluation of both the set of multi-tenant architectures and decision criteria is finalized, it is known how these artifacts look. The decision support model can be accommodated for this. It is presented in Figure 23. The first phase, now named with the more descriptive title *Decision Criteria Assessment*, is given actual steps that need to be performed in this phase and the used artifact is described too. The steps are based on the third activity in Figure 5 in which goals are established on which the decision criteria should be based. The set of criteria identified and evaluated in this research may not completely match the interest of a SaaS provider, i.e. the set lacks certain factors that are of interest to the decision makers to evaluate on or includes criteria not of interest to the decision makers. Therefore, prior to the calculation phase, this set of attributes needs to be assessed on the completeness and minimum size property. These steps are now displayed in the first phase. The used artifact during this assessment is the criteria set.

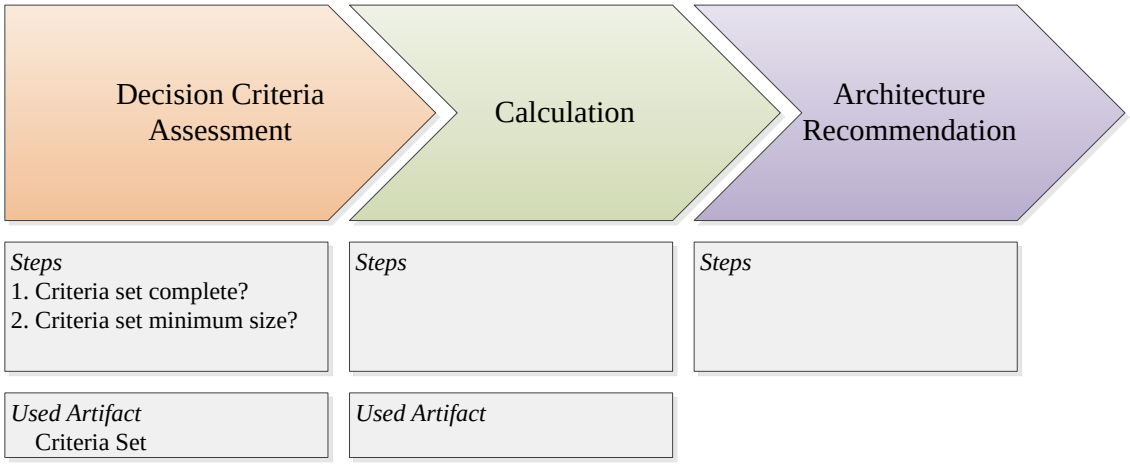



Figure 23: Multi-Tenant Architecture Selection Model

Table 18: Frequencies of Chosen Likert Items on the Discrimination and Deciding Factor of Decision Criteria

| Decision Criterion | Discrimination | | | | | | | | Deciding Factor | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | $\mu_{\frac{1}{2}}$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | $\mu_{\frac{1}{2}}$ |
| Performance Efficiency | 0 | 0 | 0 | 0 | 5 | 4 | 1 | 5.5 | 0 | 1 | 0 | 0 | 2 | 5 | 1 | 6.0 |
| Time Behavior | 0 | 0 | 0 | 2 | 4 | 3 | 1 | 5.0 | 0 | 0 | 0 | 1 | 3 | 4 | 1 | 6.0 |
| Resource Utilization | 0 | 0 | 0 | 4 | 4 | 2 | 0 | 5.0 | 0 | 2 | 1 | 1 | 2 | 3 | 0 | 5.0 |
| Capacity | 1 | 0 | 1 | 0 | 5 | 3 | 0 | 5.0 | 1 | 0 | 0 | 0 | 5 | 3 | 0 | 5.0 |
| Throughput | 1 | 0 | 4 | 0 | 3 | 2 | 0 | 4.0 | 1 | 0 | 2 | 2 | 2 | 2 | 0 | 4.0 |
| Number of Tenants | 1 | 0 | 0 | 0 | 4 | 4 | 1 | 5.5 | 1 | 0 | 0 | 0 | 3 | 3 | 2 | 6.0 |
| Number of End-users | 1 | 0 | 1 | 1 | 3 | 3 | 1 | 5.0 | 1 | 0 | 0 | 0 | 2 | 4 | 2 | 6.0 |
| Reliability | 0 | 0 | 2 | 2 | 2 | 4 | 0 | 5.0 | 0 | 0 | 1 | 1 | 3 | 3 | 1 | 5.0 |
| Availability | 0 | 1 | 1 | 2 | 2 | 3 | 1 | 5.0 | 0 | 1 | 0 | 1 | 2 | 3 | 2 | 6.0 |
| Recoverability | 1 | 1 | 0 | 1 | 4 | 2 | 1 | 5.0 | 0 | 0 | 0 | 0 | 5 | 3 | 1 | 5.0 |
| Security | 0 | 0 | 0 | 0 | 2 | 5 | 3 | 6.0 | 0 | 0 | 0 | 0 | 1 | 4 | 4 | 6.0 |
| Confidentiality | 0 | 0 | 0 | 0 | 4 | 4 | 2 | 6.0 | 0 | 0 | 0 | 0 | 2 | 3 | 4 | 6.0 |
| Integrity | 1 | 0 | 1 | 2 | 3 | 2 | 1 | 5.0 | 0 | 0 | 1 | 2 | 2 | 2 | 2 | 5.0 |
| Authenticity | 1 | 2 | 3 | 0 | 2 | 1 | 1 | 3.0 | 0 | 0 | 3 | 1 | 1 | 2 | 2 | 5.0 |
| Maintainability | 0 | 0 | 0 | 1 | 3 | 5 | 1 | 6.0 | 0 | 0 | 0 | 1 | 1 | 7 | 0 | 6.0 |
| Portability | 0 | 0 | 2 | 4 | 4 | 0 | 0 | 4.0 | 0 | 0 | 3 | 3 | 2 | 1 | 0 | 4.0 |
| Deployment Time | 0 | 0 | 0 | 2 | 5 | 3 | 0 | 5.0 | 0 | 0 | 0 | 2 | 4 | 3 | 0 | 5.0 |
| Flexibility | 0 | 0 | 0 | 4 | 2 | 3 | 1 | 5.0 | 0 | 0 | 1 | 1 | 3 | 3 | 1 | 5.0 |
| Variability | 0 | 0 | 1 | 3 | 2 | 3 | 1 | 5.0 | 0 | 0 | 1 | 4 | 3 | 0 | 1 | 4.0 |
| Diverse SLA | 0 | 0 | 1 | 4 | 2 | 3 | 0 | 4.5 | 0 | 0 | 2 | 5 | 1 | 1 | 0 | 4.0 |
| Software Complexity | 2 | 1 | 0 | 0 | 2 | 3 | 2 | 5.5 | 1 | 0 | 2 | 0 | 1 | 4 | 1 | 6.0 |
| Monitoring | 0 | 0 | 1 | 0 | 5 | 4 | 0 | 5.0 | 0 | 0 | 1 | 1 | 4 | 3 | 0 | 5.0 |

# 7 Decision Making Method Selection

A decision making process consists of multiple steps, see Figure 5. After the identification and evaluation of both the alternatives and decision criteria, a decision making method needs to be selected. This selection is an important choice and there exist a multitude of decision making methods. Each method has its drawbacks and benefits and as a result one is more suitable in a specific context than another. The method selection depends for a large part on the complexity of the decision problem. A complex method can unnecessarily complicate a simple problem.

As already mentioned in Section 3.2.1, the decision problem this research addresses falls in the domain of Multiple Attribute Decision Making (MADM). Therefore, this research should use a decision making method based on MADM. There exist many different MADM methods, but they share certain aspects too. An example in which the methods differ is how the performance values are processed to rank the alternatives. A couple of widely used MADM methods are discussed here. Go to Section 7.5 to go straight to the selected decision making method and the argumentation for this selection.

## 7.1 Elementary Methods

Elementary decision methods are relative simple approaches and require no mathematical computations. They are best suitable for decision problems with few alternatives and criteria and a single decision maker. They can be implemented rapidly. Examples of elementary decision methods are pros and cons analysis (Baker et al., 2001), maximin and maximax methods, conjunctive and disjunctive methods, and lexicographic methods (Linkov et al., 2005).

## 7.2 Cost-Benefit Analysis

In a cost and benefit analysis (CBA) the net present values (NPV) of competing investments or projects are calculated (Layard & Glaister, 1994). The NPV of an investment defines how much value can be added. This value is calculated by first assigning monetary values to costs and benefits in each year of the program. Then, these costs and benefits in future years are discounted back to the present value. Finally, they are summed. Projects with a negative NPV should be rejected, those with a positive value can be accepted. CBA is unsuitable when benefits and costs exist that are unable to be quantified in monetary terms, but is a popular tool for guiding public policy (Dodgson, Spackman, Pearman, & Phillips, 2009).

## 7.3 Multi-Attribute Utility Theory Methods

The basis of Multi-Attribute Utility Theory (MAUT) methods is the use of utility functions (Baker et al., 2001). The performance values of the alternatives against the decision criteria can be quantitative or qualitative. Quantitative values are objective and obtained from facts, qualitative values are subjective and judgmental. Utility functions are useful, because they transform both qualitative and quantitative performance values to a common dimensionless scale (Fülöp, 2005). This ensures the weights reflect the relative importance of the criteria properly. Furthermore, utility functions convert performance values so a higher preference corresponds to a higher utility value. Finally, usually a normalization process takes place on non-negative rows in the decision matrix. Some popular MAUT are now described in more detail.

### 7.3.1 Weighted Sum Model

The most commonly used approach is the weighted sum model (WSM), also called simple additive weighting (SAW) (Zanakis, Solomon, Wishart, & Dublish, 1998). In case there are $N$ alternatives and $M$ criteria, the best alternative is calculated using the following expression (Fishburn, 1967):

$$A_{WSM} = \max_j \sum_{i=1}^{M} a_{ij} w_i, \quad \text{for } j = 1, 2, ..., N \tag{1}$$

where $A_{WSM}$ is the total performance value of the best alternative. This model assumes that the total performance value of each alternative is defined by the sum of the products given in Equation (1). When the units of the performance values are the same, i.e. in single-dimensional cases, WSM can be used without difficulty, but problems arise when it is applied in multi-dimensional problems (Triantaphyllou, Shu, Sanchez, & Ray, 1998). The assumption will then be violated.

### 7.3.2 Weighted Product Model

Another technique is the weighted product model (WPM). Very similar to the WSM approach, it uses multiplication instead of addition. The method considers a comparison between two alternatives. When alternatives $A_k$ and $A_l$ are compared, their ratio is calculated according to the following expression (Bridgman, 1922):

$$R(A_k/A_l) = \prod_{i=1}^{M} (a_{ik}/a_{il})^{w_i} \tag{2}$$

If $R(A_k/A_l) > 1$, then $A_k$ is better than $A_l$, in the case of maximization. The best alternative is the one where all ratios with each other alternative are greater than or at least equal to one. WPM can be used in both single- and multi-dimensional problems, because its structure eliminates any units of measure (Triantaphyllou et al., 1998).

### 7.3.3 Analytic Hierarchy Process

The Analytic Hierarchy Process (AHP) is a systematic procedure in which the decision problem, the decision criteria and the alternatives are hierarchically represented (Saaty, 1990). The top of the hierarchy states the goal of the decision, criteria are structured in intermediate levels, and the alternatives are in the lowest level of the hierarchy. By use of a series of pairwise comparison judgments, in which elements on equal level are compared with each other with respect to the element immediately above it, the relative strength or intensity of the elements can be expressed. Comparisons between criteria represent the relative importance of one criterion over another. This way, a decision maker expresses its interests over the various criteria. Comparisons between alternatives represent the relative preference of one alternative over another. These comparisons regularly use the following nine-point scale of absolute judgments:

1 = equal importance or preference
3 = moderate importance or preference
5 = strong importance or preference
7 = very strong or demonstrated importance or preference
9 = extreme importance or preference

There exist two measurement options in AHP, relative and absolute (Saaty, 1990). Both measurements use paired comparisons to obtain priorities for the criteria with respect to the goal. In the relative measurement, paired comparisons are performed throughout the hierarchy in each level with respect to the level immediately above. In absolute measurement, paired comparisons are also performed throughout the hierarchy except at the lowest level, the one of the alternatives. Instead, the alternatives are ranked in terms of rating intensities or standards for each criterion. The intensities define variations of a criterion so the performance of alternatives for that criterion can be classified. The intensities can be expressed as numerical values when the criterion is measurable, or in qualitative terms such as A, B, C, D, E, and F (Saaty, 1994). In addition, the intensities themselves of each criterion are pairwise compared just like the other paired comparisons. The advantage of the relative measurement is that it's more accurate. The absolute

measurement has the advantage that a decision problem with of a large number of alternatives can be solved more quickly (Saaty, 2008).

The comparisons are arranged in comparison matrices and from each matrix the priority vector is calculated. The priority vector defines the best fit to the judgments expressed in that matrix (Fülöp, 2005). One approach to calculate this, is the logarithmic least squares method. In this method, the geometric mean of each row in the matrix is calculated and each geometric mean is divided by the sum of all geometric means in that matrix (Saaty & Vargas, 1984). The individual priorities are used to weigh the priorities in the level immediately below. This is done for every element. The final priorities of the alternatives are obtained by adding these weighed values. The alternative with the highest final priority is most preferred.

## 7.4 Outranking Methods

In outranking methods, alternatives are systematically compared on each criterion. It is based on the principle that alternatives can dominate one another (Kangas, Kangas, Leskinen, & Pykäläinen, 2001). Roy (1996) defined that an alternative outranks another alternative if it performs better on some criteria and at least as well on all other. An alternative is said to be dominated when it scores less on some criteria and not better than equal on all others. According to Linkov et al., "Outranking models are appropriate when criteria metrics are not easily aggregated, measurement scales vary over wide ranges, and units are incommensurate or incomparable" (2005, p. 5). Well-known outranking methods are ELECTRE (Roy, 1990) and PROMETHEE (Vincke & Brans, 1985).

## 7.5 Selection of AHP Decision Making Method

The decision making method applied in this research is the absolute measurement of AHP. The elementary decision making methods are insufficiently extensive to represent the complex decision problem addressed in this research. The outranking methods require quantitative data to determine the best alternative. No sufficient quantitative data of the identified criteria exists to rank the multi-tenant architectures. All quantitative data found was a study of Z. Wang et al. (2008) who evaluated the performance of various isolation patterns in the data tier in terms of the transactions per second and the active tenant number. Contrarily, the multi-tenant architectures in this research are a combination of isolation patterns in both the data tier and the application tier. Qualitative data, on the other hand, is available and AHP can handle this type of data very well, because subjective judgments are translated to performance scores.

Furthermore, weighing the relative importance of the decision criteria is an extensive process within AHP. This is an important aspect, because the values of these weights are assigned by a SaaS provider and need to represent its interests. SaaS providers with different interests will thus define a different set of weights for the decision criteria. These weights eventually result in the best multi-tenant architecture for that service provider. It's therefore of extreme importance a decision method is chosen that incorporates this aspect thoroughly. The analytic hierarchy process does this.

### 7.5.1 Decision Hierarchy

With the selection of AHP as the decision making method, the decision hierarchy depicting the goal, criteria and sub-criteria, and architectures can be structured, see Figure 24. The first level shows the goal of the decision problem, selecting the best multi-tenant architecture. The second level consists of the decision criteria, some of which divided in sub-criteria, which make up the third level of the hierarchy. The sub-criteria capacity is categorized even further and these define the fourth level. In the lowest level the alternatives, the multi-tenant architectures named MTA 1 to MTA 12, reside.

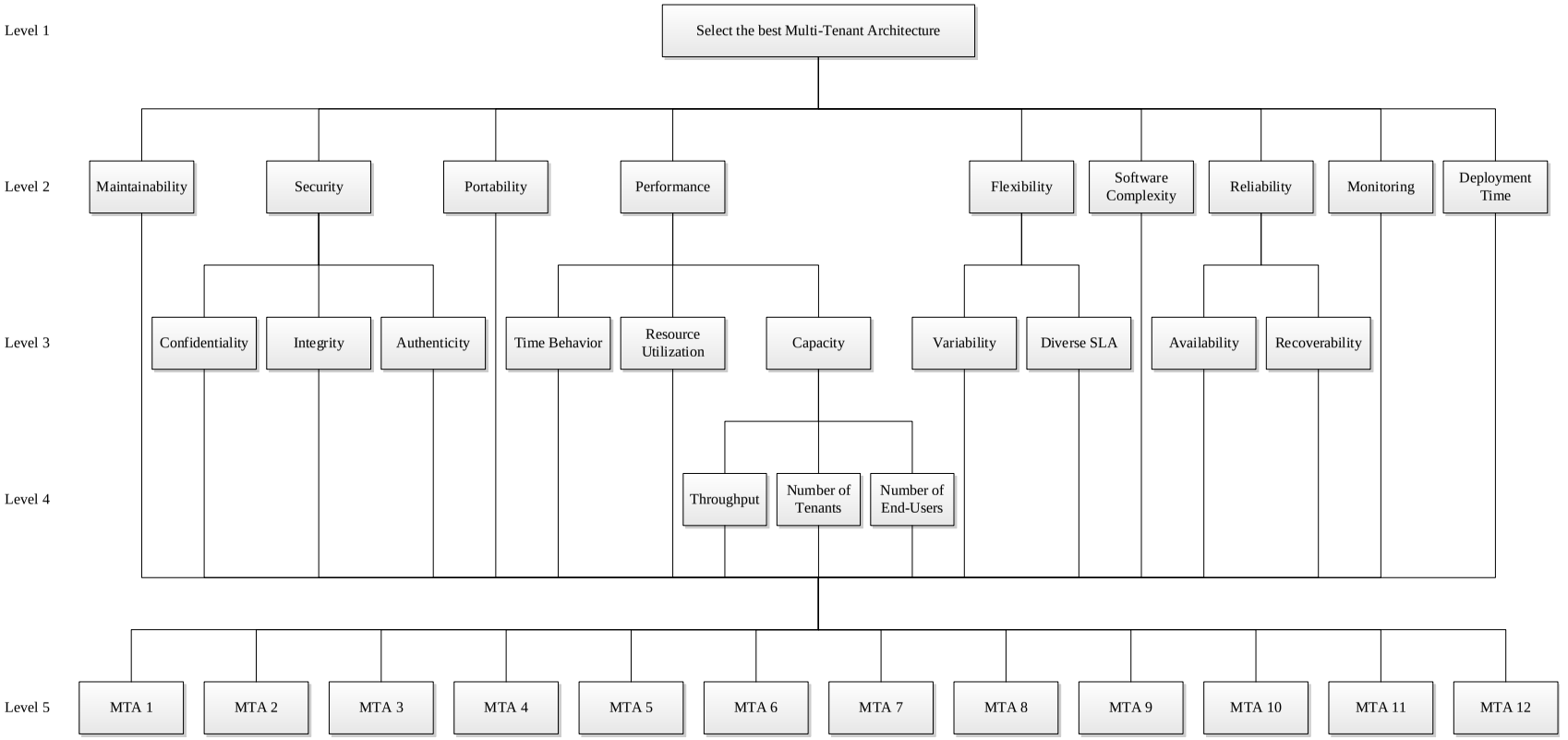



Figure 24: Hierarchy of the Multi-Tenant Architecture Decision Problem

### 7.5.2 Selection Between Measurement Approaches

Next is the argumentation of choosing one of the two measurement approaches in AHP. The relative and absolute measurement can differ greatly on the number of values to be determined. This measure directly relates to the amount of time and effort necessary to solve a decision problem. Both measurements only differ in their approach at the bottom level of the hierarchy. So by calculating the number of values to be determined in the bottom level of the hierarchy, the difference in cognitive load of both measurements can be expressed. In relative measurement, this number $V_R$ equals to the amount of pairwise comparisons $PC_R$ performed. In absolute measurement, this number $V_A$ equals to the summation of the amount of comparisons $PC_A$ and the amount of rankings $R$ performed. For the relative measurement, the amount of pairwise comparisons $PC_R$ performed in the bottom level of the hierarchy depends on the number of alternatives $N$ and criteria $M$ and is calculated as follows:

$$V_R = PC_R = \frac{1}{2}(N^2 - N)M \tag{3}$$

For this research, where $N = 12$ and $M = 17$, $PC_R$ equals to 1122 comparisons and thus $V_R$ equals to 1122 number of values. For the absolute measurement applies:

$$V_A = PC_A + R \tag{4}$$

In absolute measurement, $PC_A$ is defined as the sum of all of the number of pairwise comparisons of the intensity levels for each criterion. To simplify, the number of intensity levels $I$ is identical for each criterion. Then, with the number of criteria $M$, $PC_A$ is calculated as follows:

$$PC_A = \frac{1}{2}(I^2 - I)M \tag{5}$$

The number of rankings $R$ is calculated as:

$$R = M \times N \tag{6}$$

where $N$ is the number of alternatives. For this research, where $N = 12$, $M = 17$ and if $I$ is set at 5, $V_A = 374$.

As calculated, the relative measurement requires exactly three times more values to be determined in the bottom level of the hierarchy than the absolute measurement. Also, all $PC_A$ comparisons need to be provided by the decision makers. These comparisons define the relative importance of the intensity levels, thus partly represent the interest of a SaaS provider and are therefore unable to be defined by experts. So in this research, the actual amount of values to be determined through absolute measurement, $V_A$ is just $R$ which is 204. Relative to the relative measurement, the absolute approach requires significant less values to be determined. Therefore, absolute measurement is chosen as the decision making method in this research.

The following pairwise comparison matrices exist for the hierarchy in Figure 24 and with the selection of the absolute measurement approach. One for the criteria in the second level with respect to the goal. Four matrices in the third level: one for the sub-criteria under security, one for the sub-criteria under performance, one for the sub-criteria under flexibility and finally one for the sub-criteria under reliability. Then there is another comparison matrix in the fourth level for the sub-sub-criteria under capacity. Finally, the intensity levels, which define a classification of the range of performance, need to be pairwise compared on preference, just like the criteria and sub-criteria. The intensity for each criterion is expressed in the same qualitative terms. The relative preference among these intensity levels is equal under different criteria, i.e. the degree of preference of one intensity level to another intensity level under a specific criterion is the same degree under a different criterion. This approach is used in various decision problems (Islam & Rasad, 2006; C. Yang & Huang, 2000). Therefore, one comparison matrix exist for the intensity levels bringing the total number of comparison matrices to seven.

## 7.6 Accommodation of the Multi-Tenant Architecture Selection Model

The decision support model can now be accommodated for the selection of the decision making method. It is presented in Figure 25. The second phase is given a more descriptive name - *Priority Calculation* - and the required steps in this phase are added. The decision matrix is the used artifact in this phase, the creation of this artifact is the subject of the following section.

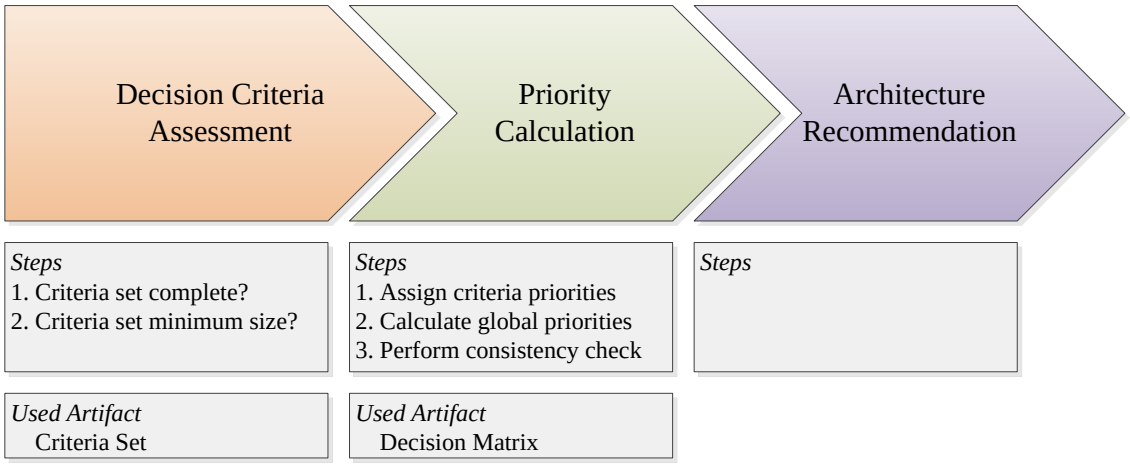



Figure 25: Multi-Tenant Architecture Selection Model

# 8 Ratings of Multi-tenant Architectures on Criteria

In absolute measurement of AHP, the pairwise comparisons between the criteria and between the intensity levels define the relative level of importance for the decision makers. These pairwise comparisons are thus specific to a SaaS provider and define the interest of that SaaS provider. This interest is what defines the most suitable multi-tenant architecture for a particular SaaS provider and depends on these pairwise comparisons.

How the multi-tenant architectures perform on the decision criteria is not specific to a SaaS provider and thus can be determined in advance. As explained in the previous section, these performance values are based on qualitative data.

The covering criteria, i.e. criteria not divided in further sub-criteria, are categorized into intensity levels or standards. There exist no pre-defined intensity levels for the decision criteria identified in this research. In addition, the criteria can not be expressed in single measurable units. The intensity levels are therefore expressed in qualitative terms. This is done for each covering criterion.

## 8.1 Strategy

The qualitative data is obtained in a similar manner as the expert evaluations of the multi-tenant architectures and criteria in Section 6. Domain experts are asked to provide their judgments on the performance of the multi-tenant architectures on the criteria.

### 8.1.1 Instrument

A series of questionnaires is used to determine the performance values. They are administered on-site by an evaluator. Therefore, questions by the experts can be answered and clarification can be provided when needed. The language used between the evaluator and experts is Dutch, because it is their native language. The experts complete the questionnaire on paper. An English translation can be found in Appendix E.

### 8.1.2 Experts

The experts involved in the questionnaire are the same experts involved in the previous evaluation of the multi-tenant architectures and the decision criteria. This was decided, because those experts were already familiar with this research due to that earlier evaluation. One expert mentioned in that evaluation that he was unable to answer the questions on the decision criteria with respect to the deciding factor. Another expert from the previous questionnaire declared he would not participate in a second one. Those experts were not involved in this second questionnaire. Although this means two experts fewer are involved, the amount of eight experts is considered sufficient to generalize carefully in the domain of multi-tenant architectures. Even though this sample size is relatively low, prior research shows that decision problems addressed with AHP not necessarily require a large sample size (Lam & Zhao, 1998).

### 8.1.3 Format

This second on-site administered questionnaire starts with a short introduction referring to the previous questionnaire that all multi-tenant architectures and decision criteria passed the evaluation and will be included in the Multi-Tenant Architecture Selection Model. Then, the questionnaire itself is handed over together with the schematics of the multi-tenant architectures, Figures 10 to 21. The architectures in the questionnaire are referenced using the same numbering used in this research. In the questionnaire, each expert is asked to rank the multi-tenant architectures against the decision criteria in terms of intensity levels. The intensity levels for all covering criteria are expressed in five qualitative terms: poor, below average, average, good and excellent. This 5-point Likert scale is used, because it provides sufficient grades of intensity levels.

In the previous questionnaire, experts were asked to define to what extent they found the multi-tenant architectures to be feasible. It's possible that if an expert defined an architecture as not or weakly feasible in this previous questionnaire, he is not able to carefully judge the performance of that architecture. Therefore, at the beginning of each ratings questionnaire, the expert is told it's not necessary to rate those architectures that were defined as not or weakly feasible by him. The expert is allowed to rate those architectures, if he finds himself comfortable to do so. The experts need to rate the architectures on all decision criteria.

### 8.1.4 Analysis Procedure

Expert ratings from an architecture on a criterion are aggregated and, equivalent as in the first questionnaire, the final score is calculated as the median. A total of eight experts participated in rating the architectures, so there are eight ratings for each architecture on a single criterion. Inherent to the median, this means it is possible the median is located between two adjacent qualitative terms.

In addition to the median, the discrimination factor is described. The discrimination factor of a criterion can be calculated numerically as the variance of the performance scores for that criterion. The variance of a set of numbers describes how far these values lie from the mean and is usually denoted as $\sigma^2$. If all scores are equal, the variance is 0. Maximum variance is 4.4 and achieved when half of the architectures is rated with the lowest possible rating (1), and the other half is rated with the maximum rating (5). This high degree of variance is not expected. For each multi-tenant architecture there exists another architecture that only differs to a small extent. The performance ratings between these two architectures probably won't differ much too. This causes the discrimination factors of the criteria to be significant lower than the maximum variance. However, the differences between these discrimination factors can be of interest.

## 8.2 Findings

Most experts completed the questionnaire in roughly one hour. There were more questions asked about the criteria during this questionnaire than during the first questionnaire. This is noteworthy, because all criteria in the second questionnaire are described in the first questionnaire too. The reason for this might be that in the second questionnaire experts need to judge the actual performance of the architectures on the criteria and the first questionnaire only asks if a criterion discriminates among the architectures. The second questionnaire requires a more thorough understanding of the criteria to make judgments.

The question about the decision criteria *number of tenants* in particular received many questions. Most experts experienced indistinctness about the aspect of scalability, declaring they view scalability as the complexity to offer the application to extra tenants, with a high scalability translating to a low complexity and vice versa. The viewpoint of scalability in this work however, is the extent of extra resources necessary to offer the application to extra tenants, with a high scalability translating to little extra resources necessary and vice versa. After three consecutive experts indicated this lack of clarity, this view was clarified in advance to the subsequent experts.

## 8.3 Results

This section presents the final decision matrix. It contains all previous identified multi-tenant architectures, decision criteria, and the performance values of the architectures with respect to the criteria. It is displayed in Table 19. All columns, except the last one, represent the multi-tenant architectures. The final column displays the discrimination factor as the variance, denoted as $\sigma^2$, of the ratings. Decision criteria are illustrated in the rows of the table. The ratings, described in terms of semantical qualitative terms, are transformed to numerical values for calculation. The highest qualitative term, *excellent*, corresponds to the value 5, the lowest qualitative term, *poor*, to the value 1. The values in the table define the performance scores of the architectures against

the decision criteria. A performance score denotes the median of the series of ratings experts gave to the matching architecture on the corresponding criterion.

For all the individual ratings provided by the experts, see Tables 25 to 41 in Appendix F. The median and standard deviation is included there too. The standard deviation is a measure to show the extent of agreement among the experts. A low standard deviation translates to a high agreement and vice versa. Sets of ratings with a low agreement need to be considered with more caution.

Table 19: Ratings of the MTA's Against the Decision Criteria

| | MTA 1 Dedicated AS & Dedicated DBS | MTA 2 Shared AS & Dedicated DBS | MTA 3 Shared Instance & Dedicated DBS | MTA 4 Dedicated AS & Shared DBS | MTA 5 Shared AS & Shared DBS | MTA 6 Shared Instance & Shared DBS | MTA 7 Dedicated AS & Shared DB | MTA 8 Shared AS & Shared DB | MTA 9 Shared Instance & Shared DB | MTA 10 Dedicated AS & Shared Schema | MTA 11 Shared AS & Shared Schema | MTA 12 Shared Instance & Shared Schema | Discrimination Factor ($\sigma^2$) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Time Behavior | 5.0 | 4.0 | 4.0 | 4.0 | 4.0 | 3.0 | 4.0 | 3.5 | 3.0 | 3.5 | 3.0 | 2.0 | 0.6 |
| Resource Utilization | 2.5 | 2.5 | 3.0 | 2.5 | 3.0 | 3.0 | 3.0 | 3.0 | 4.0 | 3.0 | 3.0 | 4.5 | 0.4 |
| Throughput | 4.5 | 3.0 | 3.0 | 4.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 0.2 |
| Number of Tenants | 1.0 | 3.0 | 3.0 | 3.0 | 3.5 | 4.0 | 3.0 | 4.0 | 4.0 | 3.0 | 4.0 | 5.0 | 1.0 |
| Number of End-Users | 2.5 | 3.5 | 3.0 | 3.0 | 3.5 | 3.5 | 3.0 | 3.5 | 4.0 | 3.5 | 4.0 | 4.0 | 0.2 |
| Availability | 4.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 0.1 |
| Recoverability | 5.0 | 4.5 | 4.5 | 4.0 | 4.0 | 4.0 | 3.0 | 3.0 | 3.0 | 2.0 | 2.0 | 2.0 | 1.1 |
| Confidentiality | 5.0 | 4.5 | 4.0 | 4.0 | 4.0 | 4.0 | 3.5 | 3.0 | 3.0 | 2.0 | 2.0 | 2.0 | 1.0 |
| Integrity | 4.5 | 4.0 | 3.0 | 4.0 | 3.5 | 3.0 | 3.5 | 3.0 | 3.0 | 3.0 | 2.5 | 2.5 | 0.4 |
| Authenticity | 4.5 | 3.5 | 3.0 | 3.5 | 3.0 | 3.0 | 4.0 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 | 0.2 |
| Migration | 5.0 | 5.0 | 5.0 | 4.5 | 4.5 | 4.5 | 4.0 | 4.0 | 4.0 | 3.0 | 3.0 | 3.0 | 0.6 |
| Deployment Time | 1.5 | 3.0 | 3.0 | 2.5 | 3.5 | 4.0 | 3.0 | 4.0 | 4.0 | 3.0 | 4.0 | 5.0 | 0.8 |
| Variability | 5.0 | 4.0 | 2.0 | 5.0 | 4.0 | 2.0 | 4.5 | 3.5 | 2.0 | 2.5 | 2.0 | 1.0 | 1.9 |
| Diverse SLA | 5.0 | 4.0 | 3.0 | 4.0 | 3.5 | 2.5 | 4.0 | 3.0 | 3.0 | 3.0 | 2.5 | 2.0 | 0.7 |
| Software Complexity | 5.0 | 4.5 | 4.0 | 4.5 | 4.5 | 3.5 | 4.0 | 4.0 | 3.0 | 2.5 | 2.5 | 2.0 | 0.9 |
| Monitoring | 1.0 | 2.5 | 3.0 | 2.5 | 3.0 | 3.0 | 3.0 | 4.0 | 4.0 | 3.0 | 4.0 | 5.0 | 1.0 |
| Maintainability | 1.5 | 2.5 | 3.0 | 2.0 | 3.0 | 3.5 | 2.5 | 4.0 | 4.0 | 3.0 | 4.0 | 5.0 | 1.0 |

A small consistency check of the data provided by the experts can be carried out. In the first questionnaire experts are asked to express their opinion on the extent of discrimination of each criterion among the multi-tenant architectures. In the second questionnaire they need to actually appoint ratings to the architectures with respect to the criteria. Therefore, it is assumed that experts defining a criterion as having no or a weak discriminating factor, their ratings of the architectures on that criterion will lay close to on another. Several experts did define some criteria with a low discriminating factor, but this is not reflected in their second questionnaire. In contrast, some experts rated the architectures on a criterion with little variance, but defined that criterion as having a strong or very strong discriminating factor. Overall, the architectures get varied ratings on criteria.

## 8.4 Analysis

This section describes notable findings that can be concluded from the results. These findings are based on the performance related extent of shared resources in the application or data tier, or both. The discrimination factor is described as well.

**Time Behavior** Both tiers influence the performance of the time behavior aspect. The less resources shared among tenants, the better the performance will be. The discrimination factor is moderate with a 0.6 variance.

**Resource Utilization Efficiency** Again, both tiers are of influence to the performance of the resource utilization efficiency. A higher degree of multi-tenancy results in a higher efficiency. Ten of the twelve architectures receive an average or slightly below average resource utilization efficiency performance. Only MTA 9 and MTA 12 perform good and very good respectively. The discrimination factor is low ($\sigma^2 = 0.4$).

**Throughput** This criteria discriminates little ($\sigma^2 = 0.3$) among the architectures. Experts define all architectures as having a moderate throughput performance, exceptions are MTA 1 and MTA 4 with a throughput performance of very good and good respectively.

**Number of Tenants** There exists a high discrimination ($\sigma^2 = 1.0$) between the performance ratings of the architectures in respect to the scalability of the number of tenants. Both tiers influence this performance and the more resources are multi-tenant, the less extra resources are required for offering the application to extra tenants.

**Number of End-Users** Both tiers affect the performance on scaling the architectures so that a tenant can offer the application to more end-users. If more resources are shared among tenants, the scalability increases too. Discrimination is low with a variance of 0.2.

**Availability** Very little discrimination ($\sigma^2 = 0.1$) exists between the availability performance ratings. All architectures but one are rated as having a moderate performance with only MTA 1 rated as good on availability.

**Recoverability** The extent of multi-tenancy in especially the data tier affects the performance on recoverability. The deeper multi-tenancy is applied in the data tier, the less it scores on recoverability. It has a high discrimination factor with a variance of 1.1.

**Confidentiality** Also for confidentiality the performance is more affected by the extent of multi-tenancy in the data tier. Less resources shared among tenants means a higher confidentiality performance. Again a high discrimination factor exists ($\sigma^2 = 1.0$).

**Integrity** Both tiers play a role in the performance of integrity of the architectures. Overall, a higher integrity score is achieved when there are less multi-tenant resources. There is relative low discrimination ($\sigma^2 = 0.4$).

**Authenticity** The decision criterion authenticity discriminates little ($\sigma^2 = 0.2$) among the architectures. All architectures with a shared application server and shared application instance but one are rated with an average authenticity performance. The architectures with a dedicated application server and a dedicated database server or a shared database have a higher performance rated as very good and good respectively.

**Migration** Only differences of multi-tenancy in the data tier are of influence in the migration performance of multi-tenant architectures. Architectures with a shared schema perform moderate, those with a shared database good, the architectures with a shared database server very good and the architectures with a dedicated database server excellent. Discrimination is moderate with a variance of 0.6.

**Deployment Time** Both tiers affect the performance of architectures with respect to the deployment time. The deeper multi-tenancy is implement, the less deployment time is necessary. There exists are relative high discrimination ($\sigma^2 = 0.8$).

**Variability** Both tiers are of influence in the performance on variability. All architectures with a shared schema in the data tier or a shared application instance in the application tier are defined as architectures with a poor or below average variability performance. Multi-tenant architectures structured with a dedicated or shared application server and a dedicated database server or shared database server are rated with good and excellent variability scores. Variability has the highest discrimination factor with a variance of 1.9.

**Diverse SLA** Both the application tier as well as the data tier affect the how diverse the service level agreement can be offered to customers. An architecture with less multi-tenant resources results in a greater ability to offer a diverse SLA. There is moderate discrimination ($\sigma^2 = 0.7$).

**Software Complexity** The data tier is more of influence than the application tier when it comes to the software complexity of multi-tenant architectures. Little shared resources result in a lower software complexity and there exists a high discrimination with a variance of 0.9.

**Monitoring** Discrimination is high with a variance of 1.0. Both tiers affect the ease with which monitoring tasks can be executed by service providers. The deeper multi-tenancy is applied in the tiers, the higher the architectures score on monitoring performance.

**Maintainability** Again both tiers are of influence. A higher maintainability is achieved when more resources are shared among tenants. There is a high discrimination ($\sigma^2 = 1.0$).

# 9 Multi-Tenant Architecture Selection Model

This section presents the final Multi-Tenant Architecture Selection Model. It is depicted in Figure 26. It should be noted that this model should be used to *support* the decision makers of a SaaS provider in the selection of a multi-tenant architecture. It is not intended to be used as the sole decision provider. It is depicted as a roadmap consisting of three phases, in which several steps are carried out using a specific artifact. In the following subsection each phase is described in more detail.

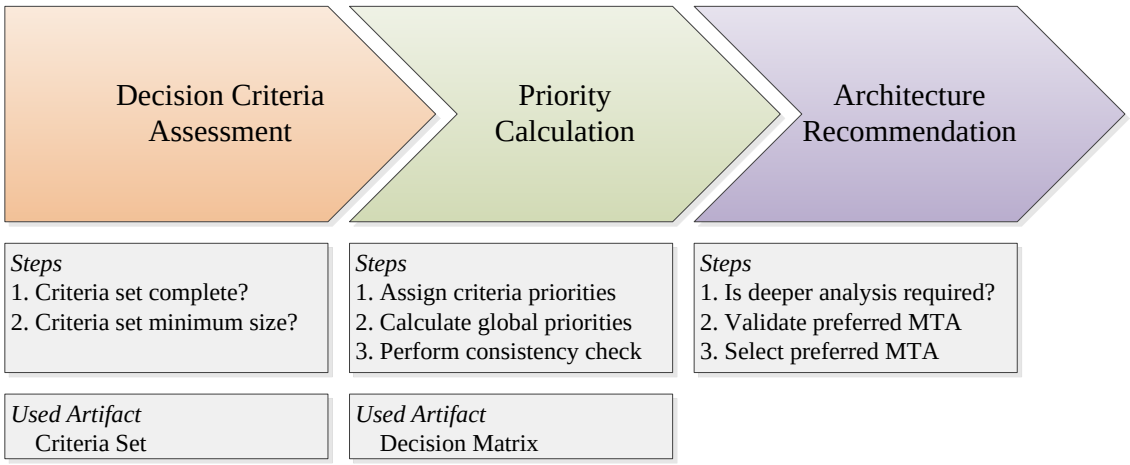



Figure 26: Multi-Tenant Architecture Selection Model

## 9.1 Decision Criteria Assessment

A SaaS provider starts the decision making process with the *Decision Criteria Assessment* phase. This phase is comprised of assessing the criteria set on completeness and minimum size. The artifact used by this phase is the criteria set, depicted in Table 16.

The first step in this phase to be undertaken by the SaaS provider, is to assess the completeness of the criteria set. This means to determine if each factor that influences the decision problem for the specific SaaS provider is covered by a criterion. If this is not the case, the decision makers of the SaaS provider can opt to add criteria by their own. In case extra criteria are added, the resulting set of criteria needs to be evaluated on the five properties that a desirable criteria list should contain. These properties are explained in Section 3.2.2. In case no criteria are added to the list, just the minimum size property should be evaluated. It may be that the criteria set contains attributes which are of no concern to the SaaS provider. If so, these criteria can be removed from the set. If the resulting criteria set differs from the one in Table 16, a new hierarchy like the one in Figure 24 can be created as well.

## 9.2 Priority Calculation

The next phase is called *Priority Calculation* in which the actual calculation using AHP takes place. First, weights, also called criteria priorities, need to be assigned to each criterion from the set resulting from the first phase. This is possible by using an absolute measurement approach in which each criterion directly receives a value lying between a predetermined range, representing the importance of that criterion. Or, using the relative measurement approach in which criteria on equal level in the hierarchy are compared with each other on relative importance with respect to their common parent.

Then, together with the decision matrix depicted in Table 19, global priorities can be calculated for each multi-tenant architecture. This requires a large number of mathematical compu-

tations. Decision support software exists that facilitates these operations. This software includes a consistency check to see if the comparison matrices completed in the previous steps contains inconsistencies.

## 9.3 Architecture Recommendation

It is possible that no multi-tenant architecture is the clear favorite but a number of multi-tenant architectures receive high priorities lying close to one another. If such a smaller set of alternatives is identified as preferred, these dominating architectures can be used in a second, more thorough, analysis. For example, Park and Hwan Lim (1999) first performed the absolute measurement of AHP on a set of five usable interfaces, and subsequently subjected the two best alternatives in a relative measurement of AHP.

Such a second analysis may entail collecting additional qualitative or evaluation data, e.g. on decision criteria defined as more important or on architectures' performance where no good agreement exists across the decision makers. All this data should be collected by the SaaS provider itself.

When there is a single preferred multi-tenant architecture, the decision support staff need to validate if this architecture in fact meets requirements, achieves the goals and results in a desired state. Finally, a recommendation report can be written and presented to the decision makers.

# 10 Discussion

The following section describes the limitations of this research. The subsequent section suggests further research.

## 10.1 Limitations

There are several limitations to the identification process of the decision criteria. Decision criteria are extracted from literature to cover a large as possible reach. This reduces bias compared to collecting criteria from a case company. Evaluation of this list is based on the extent of defined discrimination and importance to SaaS providers. However, the list of criteria is not evaluated on completeness. Experts are not asked if they thought the set lacked criteria not included in the set they evaluated. On the other hand, multi-tenant architectures are ranked on a total of 17 criteria. This is no small set and according to the minimum size principle this set should be kept as small as possible. In addition, extra criteria can be added by SaaS providers manually, but then the performance of the multi-tenant architectures with respect to these extra criteria should be defined by the SaaS provider itself too.

The minimization process of the criteria set is performed by a single researcher and not evaluated by others. It is possible that readers will not agree with all the aggregation choices made, but notable decision are described in Section 5.3.3 and the total process is illustrated in Figures 27 to 29 in Appendix C.

Also, no evaluation is conducted on the hierarchy of the decision criteria. The hierarchy is partly adopted from the quality characteristics in the software product quality model. It is assumed that this model is solidly composed of characteristics with sub-characteristics. The same arrangement is used for the hierarchy of the decision criteria identified in this research, because they resemble the characteristics in a large extent. The hierarchy of the criteria is incorporated in the questionnaire via section numbering and no questions are asked by experts about this. This does not mean that the hierarchy is evaluated however.

Another limitation of the questionnaire is the number of participants involved. The first questionnaire is completed by ten experts. In the second questionnaire eight experts participated. In addition, all experts work at the same company. This may result in bias, because all experts may work with an equal company-wide procedure. To mitigate this effect, experts are chosen that work in different business units and products. Although the number of experts is still rather small, it is assumed it is sufficient to make a conservative generalization based on their results.

Because no sufficient quantitative data yet exists, qualitative data from experts is used to rank the multi-tenant architectures on the various decision criteria. Quantitative data is more precise than subjective judgments, but results from experts can lead to very accurate results nonetheless.

There are a pair of limitations applying to the quality of the data provided by the experts. First, experts had no option in the first evaluation questionnaire to select a lack of opinion or knowledge. It is assumed that in such a case experts defined a multi-tenant architecture as not or weakly feasible. Thus, the medians of the feasibility factor of the architectures might in fact be slightly higher than this research shows. The lack of an option for *'Don't know'* can be substantiated by the fact that all participants are domain experts and therefore assume to hold sufficient knowledge. Moreover, the questionnaire is administered by an evaluator to which questions and clarifications are allowed to be asked.

Secondly, experts ranked all twelve multi-tenant architectures in the second questionnaire with respect to the decision criteria. But, some experts defined some multi-tenant architectures as not, weakly or slightly feasible in the first questionnaire. This might mean that those experts did not have complete knowledge on those multi-tenant architectures and their rankings of those multi-tenant architectures is less accurate. This research does not make any distinction between data from different experts and all data is evenly weighted.

Thirdly, some experts defined several decision criteria as having no or a weak discriminating factor. This should be reflected in their second questionnaire where multi-tenant architectures are given approximately equal ratings on those decision criteria. This is, however, not the case. Those

experts defining criteria as having no or a weak discriminating factor in the first questionnaire, evaluated the multi-tenant architectures against the corresponding criteria with varied ratings. Even more so, some experts gave the architectures on some criteria equivalent ratings in the second questionnaire, but defined those criteria as having a strong or extreme discriminating factor in the first questionnaire. This may indicate the questionnaires are not completed thoroughly consistent. A certain degree of inconsistency can always be expected when several series of surveys are to be completed by the same participants. It is probable that the experts had to think more thoroughly in the second questionnaire about consequences they had not thought of in the first questionnaire.

# 11 Conclusion

This section covers the conclusions of this research. The research questions are answered and notable findings are discussed. The main research question is formulated as follows:

RQ. *How can a SaaS provider be optimally supported in the decision process of choosing the most suitable multi-tenant architecture?*

The objective of this research is to develop a decision support model useful to all SaaS providers by supporting them in choosing the most suitable multi-tenant architecture. Many different decision methods exist, but all methods require three common elements to define the best alternative: a set of alternatives, a set of decision criteria, and performance values of the alternatives with respect to the criteria. Each of these three elements are covered by a research subquestion. The first subquestion addresses which multi-tenant architectures currently can be identified:

SQ. 1 *Which multi-tenant architectures currently exist?*

To tackle this question, first a literature study is conducted to identity levels at which multi-tenancy can be applied. From these levels, twelve multi-tenant architecture are constructed and then evaluated on feasibility by ten experts. All architectures are at least defined as being slightly feasible. The architectures are presented in Figures 10 to 21 and answer the first research subquestion. This is the first key deliverable of this research. It is of value, because it provides a complete overview of possible multi-tenancy options in the architecture that are of interest to SaaS providers.

The second subquestion focuses on identifying decision criteria:

SQ. 2 *What measurable decision criteria are of importance to SaaS providers in choosing a multi-tenant architecture and define a discrimination among these multi-tenant architectures?*

The same literature study is conducted to identify these decision criteria. First, discriminating attributes based on benefits, drawbacks, requirements and considerations related to multi-tenancy are identified from literature, resulting in a large set of attributes. This list is reduced by combining synonyms and specializations with generalizations. Infrequent attributes are excluded. The descriptions of the criteria are composed with support of quality characteristics. Then, they are evaluated on the extent of discrimination and importance to SaaS providers. No criterion is excluded based on the evaluation. The criteria are listed in Table 16 and answer the second subquestion. It forms the second key deliverable of this work and is valuable, because it describes the factors that are of influence when evaluating multi-tenancy options. It supports SaaS providers, because it forms a list of criteria that need to be considered in the decision making process. Also, it provides SaaS providers with insight what criteria are more important that other criteria.

Prior to ranking the architectures in respect to the criteria, a decision making method is selected. The analytic hierarchy process with absolute measurement is chosen, because it handles qualitative data very well, incorporates an extensive process for defined weights for the decision criteria, and is less time-consuming than the relative measurement approach.

The final research subquestion is directed at finding performance ratings of the architectures on the criteria:

SQ. 3 *How do the multi-tenant architectures perform on each decision criterion?*

Eight experts are asked to rank each multi-tenant architecture on each criterion on a scale of one to five. The medians of all these rankings represent the performance ratings of the architectures against the criteria and are shown in Table 19. It answers the final research subquestion and is the third key deliverable of this research. Its value lies in the combination of the set of multi-tenant architectures and the set of decision criteria. It brings these sets together and shows the

relation between them. It provides insight in the strengths and weaknesses of the multi-tenant architectures and is essential to execute the decision making process.

With the identification of these three key deliverables, the analytic hierarchy process decision making method can be carried out. The deliverables are part of an overall Multi-Tenant Architecture Selection Model, described and explained in Section 9. With this model, decision makers are supported in the process of selecting a multi-tenant architecture. It describes the steps to be carried out and the key deliverables used. The model will save effort, time and potential problems in the future for SaaS providers. The Multi-Tenant Architecture Selection Model is the most valuable deliverable of this research. It puts all previous mentioned key deliverables in a coherent whole.

In the first phase of the model, SaaS providers need to assess the decision criteria set on completeness and minimum size. Then, in the second phase, decision makers determine the weight of each criterion. The subsequent step is the actual calculation of the preference of each multi-tenant architecture. Decision making software packages exist that facilitates these calculations. The final phase covers an evaluation to verify if deeper analysis is necessary and validate the most preferred architecture.

In addition to the actual decision support model, this research is relevant for the scientific community. This work fills a gap in literature by evaluating multi-tenant architectures in the application and data layer as a whole.

## 11.1 Further Research

This work is the first step of using decision making theory for choosing the most suitable multi-tenant architecture. For this purpose, twelve architectures are constructed covering each possible arrangement of resources with regard to the application and data tier. In addition, 22 decision criteria are defined on which the architectures are rated. No demonstration and evaluation of these ratings is conducted in this research. It is suggested further research should focus on demonstrating the analytic hierarchy process in conjunction with the decision matrix in Table 19 at several companies. Then, the ratings can be evaluated resulting in possible adjustments for these performance values.

Furthermore, the ratings provided in this research are based on subjective judgments of eight experts. The accuracy of the ratings can be increased in two ways. First, a larger number of experts would decrease the standard deviation. Second, ratings are now based on the subjective judgments of experts. Qualitative data is less accurate than quantitative data. Further research should focus on collecting quantitative data for those criteria that support it. This is possible by defining measures for those criteria and evaluating the multi-tenant architectures in test setups in order to collect more objective data. These measures can be based on the quality measures used to quantify quality characteristics of the product quality model in ISO/IEC 25023 which is still under development and is based on ISO/IEC 9126-2 (ISO, 2003a) and ISO/IEC 9126-3 (ISO, 2003b).

# References


Aghera, P., Chaudhary, S., & Kumar, V. (2012). An approach to build multi-tenant saas application with monitoring and sla. In *Communication systems and network technologies (csnt), 2012 international conference on* (pp. 658–661).

Almorsy, M., Grundy, J., & Ibrahim, A. S. (2012). Tossma: A tenant-oriented saas security management architecture. In *Cloud computing (cloud), 2012 ieee 5th international conference on* (pp. 981–988).

Almutairi, A., Sarfraz, M., Basalamah, S., Aref, W., & Ghafoor, A. (2012). A distributed access control architecture for cloud computing. *Software, IEEE*, *29*(2), 36–44.

Archer, L. B. (1984). Systematic method for designers. In *Developments in design methodology* (p. 57-82). John Wiley, London.

Aulbach, S., Grust, T., Jacobs, D., Kemper, A., & Rittinger, J. (2008). Multi-tenant databases for software as a service: schema-mapping techniques. In *Proceedings of the 2008 acm sigmod international conference on management of data* (pp. 1195–1206).

Aulbach, S., Jacobs, D., Kemper, A., & Seibold, M. (2009). A comparison of flexible schemas for software as a service. In *Proceedings of the 35th sigmod international conference on management of data* (pp. 881–888).

Aulbach, S., Seibold, M., Jacobs, D., & Kemper, A. (2011). Extensibility and data sharing in evolving multi-tenant databases. In *Data engineering (icde), 2011 ieee 27th international conference on* (pp. 99–110).

Azeez, A., Perera, S., Gamage, D., Linton, R., Siriwardana, P., Leelaratne, D., ... Fremantle, P. (2010). Multi-tenant soa middleware for cloud computing. In *Cloud computing (cloud), 2010 ieee 3rd international conference on* (pp. 458–465).

Baker, D., Bridges, D., Hunter, R., Johnson, G., Krupa, J., Murphy, J., & Sorenson, K. (2001). *Guidebook to decision-making methods. developed for the department of energy* (Tech. Rep.). WSRC-IM-2002-00002.

Bakshi, K. (2011). Considerations for cloud data centers: Framework, architecture and adoption. In *Aerospace conference, 2011 ieee* (pp. 1–7).

Barker, S., Chi, Y., Moon, H., Hacigümüş, H., & Shenoy, P. (2012). Cut me some slack: latency-aware live migration for databases. In *Proceedings of the 15th international conference on extending database technology* (pp. 432–443).

Bezemer, C.-P., & Zaidman, A. (2010). Multi-tenant saas applications: Maintenance dream or nightmare? In *Proceedings of the joint ercim workshop on software evolution (evol) and international workshop on principles of software evolution (iwpse)* (pp. 88–92). New York, NY, USA: ACM.

Bezemer, C.-P., Zaidman, A., Platzbeecker, B., Hurkmans, T., & 't Hart, A. (2010). Enabling multi-tenancy: An industrial experience report. In (Vol. 0, p. 1-8). Los Alamitos, CA, USA: IEEE Computer Society.

Bobrowski, S. (2011). Optimal multitenant designs for cloud apps. In *Cloud computing (cloud), 2011 ieee international conference on* (pp. 654–659).

Bondi, A. B. (2000). Characteristics of scalability and their impact on performance. In *Proceedings of the 2nd international workshop on software and performance* (pp. 195–203).

Brassil, J. (2010). Physical layer network isolation in multi-tenant clouds. In *Distributed computing systems workshops (icdcsw), 2010 ieee 30th international conference on* (pp. 77–81).

Brereton, P., Kitchenham, B., Budgen, D., Turner, M., & Khalil, M. (2007). Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software*, *80*(4), 571–583.

Bridgman, P. (1922). *Dimensional analysis.* Yale University Press.

Cai, H., Wang, N., & Zhou, M. (2010). A transparent approach of enabling saas multi-tenancy in the cloud. In *Services (services-1), 2010 6th world congress on* (pp. 40–47).

Chong, F., & Carraro, G. (2006). Architecture strategies for catching the long tail. *MSDN Library, Microsoft Corporation*, 1–24.

Chong, F., Carraro, G., & Wolter, R. (2006). Multi-tenant data architecture. *MSDN Library, Microsoft Corporation*.

Das, S., Nishimura, S., Agrawal, D., & El Abbadi, A. (2011). Albatross: lightweight elasticity in shared storage databases for the cloud using live data migration. *Proceedings of the VLDB Endowment*, *4*(8), 494–505.

Dodgson, J., Spackman, M., Pearman, A., & Phillips, L. (2009). Multi-criteria analysis: a manual.

Domingo, E., Niño, J., Lemos, A., Lemos, M., Palacios, R., & Berbís, J. (2010). Cloudio: A cloud computing-oriented multi-tenant architecture for business information systems. In *Cloud computing (cloud), 2010 ieee 3rd international conference on* (pp. 532–533).

Dubey, A., & Wagle, D. (2007). Delivering software as a service. *The McKinsey Quarterly*, *6*, 1–12.

Eekels, J., & Roozenburg, N. (1991). A methodological comparison of the structures of scientific research and engineering design: Their similarities and differences. *Design Studies*, *12*(4), 197–203.

Elmore, A., Das, S., Agrawal, D., & El Abbadi, A. (2011). Zephyr: live migration in shared nothing databases for elastic cloud platforms. *SIGMOD (to appear)*.

Erotokritou, S., Nair, S. K., & Dimitrakos, T. (2010). An efficient secure shared storage service with fault and investigative disruption tolerance. In *Proceedings of the 2010 39th international conference on parallel processing workshops* (pp. 259–267).

Espadas, J., Molina, A., Jiménez, G., Molina, M., Ramírez, R., & Concha, D. (2011). A tenant-based resource allocation model for scaling software-as-a-service applications over cloud computing infrastructures. *Future Generation Computer Systems*.

Figueira, J., Ehrogott, M., & Greco, S. (2005). Multiple criteria decision analysis: State of the art surveys. *International Series in Operations Research & Management Science (78)*.

Fishburn, P. (1967). Additive utilities with incomplete product sets: Application to priorities and assignments. *Operations Research*, 537–542.

Foping, F., Dokas, I., Feehan, J., & Imran, S. (2009). A new hybrid schema-sharing technique for multitenant applications. In *Digital information management, 2009. icdim 2009. fourth international conference on* (pp. 1–6).

Fowler, M. (2002). *Patterns of enterprise application architecture*. Addison-Wesley Professional.

Fülöp, J. (2005). Introduction to decision making methods. *Laboratory of Operations Research and Decision Systems, Computer and Automation Institute. Hungarian: Academy of Sciences*.

Gao, B., An, W., Sun, X., Wang, Z., Fan, L., Guo, C., & Sun, W. (2011). A non-intrusive multi-tenant database software for large scale saas application. In *e-business engineering (icebe), 2011 ieee 8th international conference on* (pp. 324–328).

Ghaddar, A., Tamzalit, D., & Assaf, A. (2011). Decoupling variability management in multi-tenant saas applications. In *Service oriented system engineering (sose), 2011 ieee 6th international symposium on* (pp. 273–279).

Ghaddar, A., Tamzalit, D., Assaf, A., & Bitar, A. (2012). Variability as a service: outsourcing variability management in multi-tenant saas applications. In *Advanced information systems engineering* (pp. 175–189).

Glott, R., Husmann, E., Sadeghi, A., & Schunter, M. (2011). Trustworthy clouds underpinning the future internet. *The future internet*, 209–221.

Guo, C., Sun, W., Huang, Y., Wang, Z. H., & Gao, B. (2007). A framework for native multi-tenancy application development and management. In (Vol. 0, p. 551-558). Los Alamitos, CA, USA: IEEE Computer Society. doi: http://doi.ieeecomputersociety.org/10.1109/CEC-EEE.2007.4

Guo, C., Sun, W., Jiang, Z., Huang, Y., Gao, B., & Wang, Z. (2011). Study of software as a service support platform for small and medium businesses. *New Frontiers in Information and Software as Services*, 1–30.

Haller, K. (2011, January). Web services from a service provider perspective: tenant management services for multitenant information systems. *SIGSOFT Softw. Eng. Notes*, *36*(1), 1–4. Retrieved from `http://doi.acm.org/10.1145/1921532.1921542` doi: 10.1145/1921532.1921542

Harris, I., & Ahmed, Z. (2011). An open multi-tenant architecture to leverage smes. *European Journal of Scientific Research*, *65*(4), 601–610.

He, Q., Han, J., Yang, Y., Grundy, J., & Jin, H. (2012). Qos-driven service selection for multi-tenant saas. In *Cloud computing (cloud), 2012 ieee 5th international conference on* (pp. 566–573).

He, S., Guo, L., & Guo, Y. (2011). Elastic application container. In *Grid computing (grid), 2011 12th ieee/acm international conference on* (pp. 216–217).

He, S., Guo, L., Guo, Y., Wu, C., Ghanem, M., & Han, R. (2012). Elastic application container: A lightweight approach for cloud resource provisioning. In *Advanced information networking and applications (aina), 2012 ieee 26th international conference on* (pp. 15–22).

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004, March). Design science in information systems research. *MIS Q.*, *28*(1), 75–105.

Howard, R. (1966). Decision analysis: Applied decision theory. In *Proceedings of the fourth international conference on operations research* (pp. 55–72).

Hudli, A. V., Shivaradhya, B., & Hudli, R. V. (2009). Level-4 saas applications for healthcare industry. In *Proceedings of the 2nd bangalore annual compute conference* (p. 19).

Hui, M., Jiang, D., Li, G., & Zhou, Y. (2009). Supporting database applications as a service. In *Data engineering, 2009. icde'09. ieee 25th international conference on* (pp. 832–843).

Hwang, C., & Masud, A. (1979). *Multiple objective decision making, methods and applications: a state-of-the-art survey*. Springer-Verlag. Retrieved from `http://books.google.nl/books?id=Hz-yAAAAIAAJ`

Hwang, C., & Yoon, K. (1981). *Multiple attribute decision making: methods and applications: a state-of-the-art survey*. Springer-Verlag. Retrieved from `http://books.google.nl/books?id=X-wYAQAAIAAJ`

Islam, R., & Rasad, S. (2006). Employee performance evaluation by the ahp: A case study. *Asia Pacific Management Review*, *11*(3), 163.

ISO, J. (2003a). Iso/iec 9126-2:2003, software engineering-product quality-part 2: External metrics. *International Organization for Standardization*.

ISO, J. (2003b). Iso/iec 9126-3:2003, software engineering-product quality-part 3: Internal metrics. *International Organization for Standardization*.

ISO, J. (2011). Iso/iec 25010: 2011, systems and software engineering-systems and software quality requirements and evaluation (square)-system and software quality models. *International Organization for Standardization*.

ISO, J. (2012). Iso/iec 27000: 2012, information technology-security techniques-information security management systems-overview and vocabulary. *International Organization for Standardization*.

Jacobs, D., & Aulbach, S. (2007). Ruminations on Multi-Tenant Databases. *BTW Proceedings*, *102*, 514–521.

Jegadeesan, H., & Balasubramaniam, S. (2009). A method to support variability of enterprise services on the cloud. In *Cloud computing, 2009. cloud'09. ieee international conference on* (pp. 117–124).

Jing, J., & Zhang, J. (2010). Research on open saas software architecture based on soa. In *Computational intelligence and design (iscid), 2010 international symposium on* (Vol. 2, pp. 144–147).

Ju, L., Sengupta, B., & Roychoudhury, A. (2012). Tenant onboarding in evolving multi-tenant software-as-a-service systems. In *Web services (icws), 2012 ieee 19th international conference on* (pp. 415–422).

Kabbedijk, J., & Jansen, S. (2011). Variability in multi-tenant environments: Architectural design patterns from industry. In O. De Troyer, C. Bauzer Medeiros, R. Billen, P. Hallot, A. Simitsis, & H. Van Mingroot (Eds.), *Advances in conceptual modeling. recent developments and new directions* (Vol. 6999, p. 151-160). Springer Berlin / Heidelberg.

Kang, S., Kang, S., & Hur, S. (2011). A design of the conceptual architecture for a multitenant saas application platform. In *Computers, networks, systems and industrial engineering (cnsi), 2011 first acis/jnu international conference on* (pp. 462–467).

Kangas, J., Kangas, A., Leskinen, P., & Pykäläinen, J. (2001). Mcdm methods in strategic planning of forestry on state-owned lands in finland: applications and experiences. *Journal of Multi-Criteria Decision Analysis*, *10*(5), 257–271.

Kapuruge, M., Colman, A., & Han, J. (2011a). Achieving multi-tenanted business processes in saas applications. *Web Information System Engineering–WISE 2011*, 143–157.

Kapuruge, M., Colman, A., & Han, J. (2011b). Defining customizable business processes without compromising the maintainability in multi-tenant saas applications. In *Cloud computing (cloud), 2011 ieee international conference on* (pp. 748–749).

Keeney, R., & Raiffa, H. (1993). *Decisions with multiple objectives: preferences and value trade-offs*. Cambridge University Press.

Kitchenham, B. A., & Charters, S. (2007). *Guidelines for performing Systematic Literature Reviews in Software Engineering* (Technical Report No. EBSE-2007-01). Keele University.

Kong, L., Li, Q., & Zheng, X. (2010). A novel model supporting customization sharing in saas applications. In *Multimedia information networking and security (mines), 2010 international conference on* (pp. 225–229).

Koziolek, H. (2011). The sposad architectural style for multi-tenant software applications. In *Software architecture (wicsa), 2011 9th working ieee/ifip conference on* (pp. 320–327).

Krebs, R., Momm, C., & Konev, S. (2012). Architectural concerns in multi-tenant saas applications. In *Proceedings of the 2nd international conference on cloud computing and service science (closer12). scitepress.*

Krebs, R., Momm, C., & Kounev, S. (2012). Metrics and techniques for quantifying performance isolation in cloud environments. In *Proceedings of the 8th international acm sigsoft conference on quality of software architectures* (pp. 91–100).

Kurmus, A., Gupta, M., Pletka, R., Cachin, C., & Haas, R. (2011). A comparison of secure multi-tenancy architectures for filesystem storage clouds. *Middleware 2011*, 471–490.

Kwok, T., Laredo, J., & Maradugu, S. (2008). A web services integration to manage invoice identification, metadata extraction, storage and retrieval in a multi-tenancy saas application. In *e-business engineering, 2008. icebe'08. ieee international conference on* (pp. 359–366).

Kwok, T., & Mohindra, A. (2008). Resource calculations with constraints, and placement of tenants and instances for multi-tenant saas applications. *Service-Oriented Computing–ICSOC 2008*, 633–648.

Kwok, T., Nguyen, T., & Lam, L. (2008). A software as a service with multi-tenancy support for an electronic contract management application. In (Vol. 2, p. 179-186). Los Alamitos, CA, USA: IEEE Computer Society. doi: http://doi.ieeecomputersociety.org/10.1109/SCC.2008.138

Lam, K., & Zhao, X. (1998). An application of quality function deployment to improve the quality of teaching. *International Journal of Quality & Reliability Management*, *15*(4), 389–413.

Lang, W., Shankar, S., Patel, J. M., & Kalhan, A. (2012). Towards multi-tenant performance slos. In *Data engineering (icde), 2012 ieee 28th international conference on* (pp. 702–713).

Laplante, P. A., Zhang, J., & Voas, J. (2008). What's in a Name? Distinguishing between SaaS and SOA. *IT Professional*, *10*(3), 46–50.

Layard, R., & Glaister, S. (1994). *Cost-benefit analysis*. Cambridge University Press.

Lee, W., & Choi, M. (2012). A multi-tenant web application framework for saas. In *Cloud computing (cloud), 2012 ieee 5th international conference on* (pp. 970–971).

Li, J., Li, B., Wo, T., Hu, C., Huai, J., Liu, L., & Lam, K. (2012). Cyberguarder: A virtualization security assurance architecture for green cloud computing. *Future Generation Computer Systems*, *28*(2), 379–390.

Li, W., Zhang, Z., Wu, S., & Wu, Z. (2010). An implementation of the saas level-3 maturity model for an educational credit bank information system. In *Service sciences (icss), 2010 international conference on* (pp. 283–287).

Li, X., Liu, T., Li, Y., & Chen, Y. (2008). Spin: Service performance isolation infrastructure in multi-tenancy environment. *Service-Oriented Computing–ICSOC 2008*, 649–663.

Lin, H., Sun, K., Zhao, S., & Han, Y. (2009). Feedback-control-based performance regulation for multi-tenant applications. In (Vol. 0, p. 134-141). Los Alamitos, CA, USA: IEEE Computer Society. doi: http://doi.ieeecomputersociety.org/10.1109/ICPADS.2009.22

Linkov, I., Varghese, A., Jamil, S., Seager, T., Kiker, G., & Bridges, T. (2005). Multi-criteria decision analysis: a framework for structuring remedial decisions at contaminated sites. In *Comparative risk assessment and environmental decision making* (pp. 15–54). Springer.

Ma, D. (2007). The Business Model of "Software-As-A-Service". In *Ieee international conference on services computing (scc 2007)* (pp. 701–702). Salt Lake City, Utah, USA: IEEE Computer Society.

Ma, K., Yang, B., & Abraham, A. (2012). A template-based model transformation approach for deriving multi-tenant saas applications. *Acta Polytechnica Hungarica*, *9*(2).

Marsden, P., & Wright, J. (2010). *Handbook of survey research*. Emerald Group Pub Limited.

Mell, P., & Grance, T. (2011). The NIST Definition of Cloud Computing. *National Institute of Standards and Technology*, *53*(6).

Mietzner, R., Leymann, F., & Papazoglou, M. (2008). Defining composite configurable saas application packages using sca, variability descriptors and multi-tenancy patterns. In *Internet and web applications and services, 2008. iciw'08. third international conference on* (pp. 156–161).

Mietzner, R., Metzger, A., Leymann, F., & Pohl, K. (2009). Variability modeling to support customization and deployment of multi-tenant-aware software as a service applications. In *Proceedings of the 2009 icse workshop on principles of engineering service oriented systems* (pp. 18–25).

Mietzner, R., Unger, T., Titze, R., & Leymann, F. (2009). Combining different multi-tenancy patterns in service-oriented applications. In *Enterprise distributed object computing conference, 2009. edoc'09. ieee international* (pp. 131–140).

Momm, C., & Theilmann, W. (2011). A combined workload planning approach for multi-tenant business applications. In *Computer software and applications conference workshops (compsacw), 2011 ieee 35th annual* (pp. 255–260).

Mudigonda, J., Yalagandula, P., Mogul, J., Stiekes, B., & Pouffary, Y. (2011). Netlord: a scalable multi-tenant network architecture for virtualized datacenters. *SIGCOMM-Computer Communication Review*, *41*(4), 62.

Natis, Y. V., & Knipp, E. (2008). Reference architecture for multitenancy. *Gartner*.

Nunamaker Jr, J., & Chen, M. (1990). Systems development in information systems research. In *Twenty-third annual hawaii international conference on system sciences* (Vol. 3, pp. 631–640).

Osipov, C., Goldszmidt, G., Taylor, M., & Poddar, I. (2009). Develop and deploy multi-tenant web-delivered solutions using ibm middleware: Part 2: Approaches for enabling multi-tenancy. *IBM Corp. Website*.

Park, K., & Hwan Lim, C. (1999). A structured methodology for comparative evaluation of user interface designs using usability criteria and measures. *International Journal of Industrial Ergonomics*, *23*(5), 379–389.

Pathirage, M., Perera, S., Kumara, I., & Weerawarana, S. (2011). A multi-tenant architecture for business process executions. In *Web services (icws), 2011 ieee international conference on* (pp. 121–128).

Peffers, K., Tuunanen, T., Gengler, C., Rossi, M., Hui, W., Virtanen, V., & Bragge, J. (2006). The Design Science Research Process: A Model for Producing and Presenting Information Systems Research. In *First international conference on design science research in information systems and technology* (pp. 83–106).

Pippal, S., Sharma, V., Mishra, S., & Kushwaha, D. (2011). An efficient schema shared approach for cloud based multitenant database with authentication and authorization framework. In *P2p, parallel, grid, cloud and internet computing (3pgcic), 2011 international conference on* (pp. 213–218).

Plattner, H., Zeier, A., Plattner, H., & Zeier, A. (2011). Scaling sanssoucidb in the cloud. In *In-memory data management* (p. 193-204). Springer Berlin Heidelberg.

Preston, C., & Colman, A. (2000). Optimal number of response categories in rating scales: reliability, validity, discriminating power, and respondent preferences. *Acta psychologica*, *104*(1), 1–15.

Reinwald, B. (2010). Database support for multi-tenant applications. In *Ieee workshop on information and software as services* (Vol. 1, p. 2).

Rimal, B. P., & El-Refaey, M. A. (2010). A framework of scientific workflow management systems for multi-tenant cloud orchestration environment. In *Enabling technologies: Infrastructures for collaborative enterprises (wetice), 2010 19th ieee international workshop on* (pp. 88–93).

Rodero-Merino, L., Vaquero, L., Caron, E., Muresan, A., & Desprez, F. (2012). Building safe paas clouds: A survey on security in multitenant software platforms. *Computers & Security*.

Rossi, M., & Sein, M. (2003). Design research workshop: A proactive research approach. *Action Research*, *2005*(01.02. 2004), 1–20.

Roy, B. (1990). The outranking approach and the foundations of electre methods. In *Readings in multiple criteria decision aid* (pp. 155–183). Springer.

Roy, B. (1996). *Multicriteria methodology for decision aiding* (Vol. 12). Springer.

Ruehl, S. T., Andelfinger, U., Rausch, A., & Verclas, S. A. (2012). Toward realization of deployment variability for software-as-a-service applications. In *Cloud computing (cloud), 2012 ieee 5th international conference on* (pp. 622–629).

Sääksjärvi, M., Lassila, A., & Nordström, H. (2005). Evaluating the software as a service business model: From cpu time-sharing to online innovation sharing. In *Proceedings of the iadis international conference e-society* (pp. 177–186).

Saaty, T. (1990). How to make a decision: the analytic hierarchy process. *European journal of operational research*, *48*(1), 9–26.

Saaty, T. (1994). How to make a decision: the analytic hierarchy process. *Interfaces*, *24*(6), 19–43.

Saaty, T. (2008). Decision making with the analytic hierarchy process. *International Journal of Services Sciences*, *1*(1), 83–98.

Saaty, T., & Vargas, L. (1984). Comparison of eigenvalue, logarithmic least squares and least squares methods in estimating ratios. *Mathematical Modelling*, *5*(5), 309–324.

Sandhu, R. S., & Samarati, P. (1994). Access control: principle and practice. *Communications Magazine, IEEE*, *32*(9), 40–48.

Saripalli, P., Oldenburg, C., Walters, B., & Radheshyam, N. (2011). Implementation and usability evaluation of a cloud platform for scientific computing as a service (scaas). In *Utility and cloud computing (ucc), 2011 fourth ieee international conference on* (pp. 345–354).

Schiller, O., Schiller, B., Brodt, A., & Mitschang, B. (2011). Native support of multi-tenancy in rdbms for software as a service. In *Proceedings of the 14th international conference on extending database technology* (pp. 117–128).

Schroeter, J., Cech, S., Götz, S., Wilke, C., & Aßmann, U. (2012). Towards modeling a variable architecture for multi-tenant saas-applications. In *Proceedings of the sixth international workshop on variability modeling of software-intensive systems* (pp. 111–120).

Sengupta, B., & Roychoudhury, A. (2011). Engineering multi-tenant software-as-a-service systems. In *Icse workshop on principles of engineering service oriented systems (pesos)*.

Sun, W., Zhang, K., Chen, S.-K., Zhang, X., & Liang, H. (2007). Software as a service: An integration perspective. In B. Krmer, K.-J. Lin, & P. Narasimhan (Eds.), *Service-oriented computing icsoc 2007* (Vol. 4749, p. 558-569). Springer Berlin / Heidelberg.

Sun, W., Zhang, X., Guo, C. J., Sun, P., & Su, H. (2008, September). Software as a Service: Configuration and Customization Perspectives. In *2008 ieee congress on services part ii (services-2 2008)* (pp. 18–25). IEEE Computer Society. doi: 10.1109/SERVICES-2.2008.29

Sun, X., Gao, B., Fan, L., & An, W. (2012). A cost-effective approach to delivering analytics as a service. In *Web services (icws), 2012 ieee 19th international conference on* (pp. 512–519).

Takahashi, H., Mori, K., & Ahmad, H. F. (2010). Efficient i/o intensive multi tenant saas system using l4 level cache. In *Service oriented system engineering (sose), 2010 fifth ieee international symposium on* (pp. 222–228).

Takahashi, T., Blanc, G., Kadobayashi, Y., Fall, D., Hazeyama, H., & Matsuo, S. (2012). Enabling secure multitenancy in cloud computing: Challenges and approaches. In *Future internet communications (bcfic), 2012 2nd baltic congress on* (pp. 72–79).

Takeda, H., Veerkamp, P., & Yoshikawa, H. (1990). Modeling Design Processes. *AI Magazine*, *11*(4), 37–48.

Tang, K., Jiang, Z., Sun, W., Zhang, X., & Dong, W. (2010). Research on tenant placement based on business relations. In *e-business engineering (icebe), 2010 ieee 7th international conference on* (pp. 479–483).

Taylor, M., & Guo, C. (2007). Data integration and composite business services, part 3: Build a multi-tenant data tier with access control and security. *IBM Corporation, Armonk, NY, Dec*.

Terlecki, P., Bati, H., Galindo-Legaria, C., & Zabback, P. (2009). Filtered statistics. In *Proceedings of the 35th sigmod international conference on management of data* (pp. 897–904).

Triantaphyllou, E., Shu, B., Sanchez, S., & Ray, T. (1998). Multi-criteria decision making: an operations research approach. *Encyclopedia of electrical and electronics engineering*, *15*, 175–186.

Truyen, E., Cardozo, N., Walraven, S., Vallejos, J., Bainomugisha, E., Günther, S., ... Joosen, W. (2012). Context-oriented programming for customizable saas applications. In *Proceedings of the 27th annual acm symposium on applied computing* (pp. 418–425).

Tsai, C.-H., Ruan, Y., Sahu, S., Shaikh, A., & Shin, K. (2007). Virtualization-based techniques for enabling multi-tenant management tools. In A. Clemm, L. Granville, & R. Stadler (Eds.), *Managing virtualization of networks and services* (Vol. 4785, p. 171-182). Springer Berlin / Heidelberg.

Tsai, W.-T., Huang, Y., Bai, X., & Gao, J. (2012). Scalable architectures for saas. In *Object/component/service-oriented real-time distributed computing workshops (isorcw), 2012 15th ieee international symposium on* (pp. 112–117).

Tsai, W.-T., Huang, Y., & Shao, Q. (2011). Easysaas: A saas development framework. In *Service-oriented computing and applications (soca), 2011 ieee international conference on* (pp. 1–4).

Tsai, W.-T., Li, W., Bai, X., & Elston, J. (2011). P4-simsaas: Policy specification for multi-tendency simulation software-as-a-service model. In *Simulation conference (wsc), proceedings of the 2011 winter* (pp. 3067–3081).

Tsai, W.-T., Shao, Q., & Elston, J. (2010). Prioritizing service requests on cloud with multi-tenancy. In *e-business engineering (icebe), 2010 ieee 7th international conference on* (pp. 117–124).

Tsai, W.-T., Shao, Q., Sun, X., & Elston, J. (2010). Real-time service-oriented cloud computing. In *Services (services-1), 2010 6th world congress on* (pp. 473–478).

Tsai, W.-T., Sun, X., Shao, Q., & Qi, G. (2010). Two-tier multi-tenancy scaling and load balancing. In *e-business engineering (icebe), 2010 ieee 7th international conference on* (pp. 484–489).

Tucker, A. B. (2004). *Computer science handbook*. Chapman & Hall/CRC.

Turner, M., Budgen, D., & Brereton, P. (2003). Turning Software into a Service. *Computer*, *36*(10), 38–44.

van de Weerd, I., & Brinkkemper, S. (2009). Meta-modeling for situational analysis and design methods. In *Handbook of research on modern systems analysis and design technologies and applications* (p. 38-58). Idea Group Publishing, Hershey.

Vincke, J., & Brans, P. (1985). A preference ranking organization method. the promethee method for mcdm. *Management Science*, *31*(6), 647–656.

Waidner, M. (2009, November). *Cloud computing and security.* (Lecture Notes, Universität Stuttgart)

Walls, J., Widmeyer, G., & El Sawy, O. (1992). Building an information system design theory for vigilant eis. *Information Systems Research*, *3*(1), 36–59.

Walraven, S., Truyen, E., & Joosen, W. (2011). A middleware layer for flexible and cost-efficient multi-tenant applications. *Middleware 2011*, 370–389.

Wang, D., Zhang, Y., Zhang, B., & Liu, Y. (2009). Research and implementation of a new saas service execution mechanism with multi-tenancy support. In *Information science and engineering (icise), 2009 1st international conference on* (pp. 336–339).

Wang, H., & Zheng, Z. (2010). Software architecture driven configurability of multi-tenant saas application. *Web Information Systems and Mining*, 418–424.

Wang, M., Bandara, K. Y., & Pahl, C. (2010). Process as a service distributed multi-tenant policy-based process runtime governance. In *Services computing (scc), 2010 ieee international conference on* (pp. 578–585).

Wang, R., Zhang, Y., Liu, S., Wu, L., & Meng, X. (2011). A dependency-aware hierarchical service model for saas and cloud services. In *Services computing (scc), 2011 ieee international conference on* (pp. 480–487).

Wang, W., Huang, X., Qin, X., Zhang, W., Wei, J., & Zhong, H. (2012). Application-level cpu consumption estimation: Towards performance isolation of multi-tenancy web applications. In *Cloud computing (cloud), 2012 ieee 5th international conference on* (pp. 439–446).

Wang, Z., Guo, C., Gao, B., Sun, W., Zhang, Z., & An, W. (2008). A study and performance evaluation of the multi-tenant data tier design patterns for service oriented computing. In *e-business engineering, 2008. icebe'08. ieee international conference on* (pp. 94–101).

Watson, R., & Webster, J. (2002). Analyzing the past to prepare for the future: Writing a literature review. *Mis Quarterly*, *26*(2).

Weiping, L. (2009). An analysis of new features for workflow system in the saas software. In *Proceedings of the 2nd international conference on interaction sciences: Information technology, culture and human* (pp. 110–114).

Weissman, C., & Bobrowski, S. (2009). The design of the force. com multitenant internet application development platform. In *Proceedings of the 35th sigmod international conference on management of data* (pp. 889–896).

Wood, K., & Anderson, M. (2011). Understanding the complexity surrounding multitenancy in cloud computing. In *e-business engineering (icebe), 2011 ieee 8th international conference on* (pp. 119–124).

Xuxu, Z., Qingzhong, L., & Lanju, K. (2010). A data storage architecture supporting multi-level customization for saas. In *Web information systems and applications conference (wisa), 2010 7th* (pp. 106–109).

Yaish, H., Goyal, M., & Feuerlicht, G. (2011). An elastic multi-tenant database schema for software as a service. In *Dependable, autonomic and secure computing (dasc), 2011 ieee ninth international conference on* (pp. 737–743).

Yang, C., & Huang, J. (2000). A decision model for is outsourcing. *International Journal of Information Management*, *20*(3), 225–239.

Yang, E., Zhang, Y., Wu, L., Liu, Y., & Liu, S. (2011). A hybrid approach to placement of tenants for service-based multi-tenant saas application. In *Services computing conference (apscc), 2011 ieee asia-pacific* (pp. 124–130).

Yoon, K., & Hwang, C. (1995). *Multiple attribute decision making: an introduction* (No. 102-104). Sage Publications, Incorporated.

Yu, D., Wang, J., Hu, B., Liu, J., Zhang, X., He, K., & Zhang, L. (2011). A practical architecture of cloudification of legacy applications. In *Services (services), 2011 ieee world congress on* (pp. 17–24).

Yu, H., & Wang, D. (2011). A heuristic data allocation method for multi-tenant saas application in distributed database systems. In *Information management, innovation management and industrial engineering (iciii), 2011 international conference on* (Vol. 2, pp. 382–386).

Yuanyuan, D., Hong, N., Bingfei, W., & Lei, L. (2009). Scaling the data in multi-tenant business support system. In *Knowledge engineering and software engineering, 2009. kese'09. pacific-asia conference on* (pp. 43–46).

Zanakis, S., Solomon, A., Wishart, N., & Dublish, S. (1998). Multi-attribute decision making: A simulation comparison of select methods. *European journal of operational research*, *107*(3), 507–529.

Zhang, F., Chen, J., Chen, H., & Zang, B. (2011). Cloudvisor: Retrofitting protection of virtual machines in multi-tenant cloud with nested virtualization. In *Proceedings of the twenty-third acm symposium on operating systems principles* (pp. 203–216).

Zhang, K., Li, Q., & Shi, Y. (2011). Data privacy preservation during schema evolution for multi-tenancy applications in cloud computing. *Web Information Systems and Mining*, 376–383.

Zhang, Y., Wang, Z., Gao, B., Guo, C., Sun, W., & Li, X. (2010). An effective heuristic for on-line tenant placement problem in saas. In *Web services (icws), 2010 ieee international conference on* (pp. 425–432).

Zhou, H., Wen, Q., & Yu, X. (2011). A productive time length-based method for multi-tenancy-oriented service usage metering and billing. *Advances in Computer Science, Intelligent System and Environment*, 163–168.

Zhou, Y., Wang, Q., Wang, Z., & Wang, N. (2011). Db2mmt: a massive multi-tenant database platform for cloud computing. In *e-business engineering (icebe), 2011 ieee 8th international conference on* (pp. 335–340).

Zhou, Y. C., Liu, X. P., Wang, X. N., Xue, L., Liang, X. X., & Liang, S. (2010). Business process centric platform-as-a-service model and technologies for cloud enabled industry solutions. In *Cloud computing (cloud), 2010 ieee 3rd international conference on* (pp. 534–537).

Zimmermann, H. (1991). *Fuzzy set theory and its applications*. Allied Publishers.

# A List of Relevant Literature

Table 20: LIST OF RELEVANT LITERATURE

| Article | Article |
|---|---|
| Aghera et al., 2012 | Lin et al. (2009) |
| Almorsy, Grundy, & Ibrahim, 2012 | K. Ma, Yang, and Abraham (2012) |
| Almutairi, Sarfraz, Basalamah, Aref, & Ghafoor, 2012 | Mietzner, Leymann, and Papazoglou (2008) |
| Aulbach et al., 2008 | Mietzner, Metzger, Leymann, and Pohl (2009) |
| Aulbach, Jacobs, Kemper, & Seibold, 2009 | Mietzner, Unger, Titze, and Leymann (2009) |
| Aulbach, Seibold, Jacobs, and Kemper (2011) | Momm and Theilmann (2011) |
| Azeez et al. (2010) | Mudigonda, Yalagandula, Mogul, Stiekes, and Pouffary (2011) |
| Bakshi (2011) | Pathirage, Perera, Kumara, and Weerawarana (2011) |
| Barker, Chi, Moon, Hacigümüş, and Shenoy (2012) | Pippal et al. (2011) |
| Bezemer et al. (2010) | Plattner, Zeier, Plattner, and Zeier (2011) |
| Bezemer and Zaidman (2010) | Rimal and El-Refaey (2010) |
| Bobrowski (2011) | Rodero-Merino et al. (2012) |
| Brassil (2010) | Ruehl, Andelfinger, Rausch, and Verclas (2012) |
| Cai, Wang, and Zhou (2010) | Saripalli, Oldenburg, Walters, and Radheshyam (2011) |
| Das, Nishimura, Agrawal, and El Abbadi (2011) | Schiller, Schiller, Brodt, and Mitschang (2011) |
| Domingo et al. (2010) | Schroeter, Cech, Götz, Wilke, and Aßmann (2012) |
| Elmore, Das, Agrawal, and El Abbadi (2011) | Sengupta and Roychoudhury (2011) |
| Erotokritou, Nair, and Dimitrakos (2010) | W. Sun et al. (2008) |
| Espadas et al. (2011) | X. Sun, Gao, Fan, and An (2012) |
| Foping, Dokas, Feehan, and Imran (2009) | H. Takahashi, Mori, and Ahmad (2010) |
| Gao et al. (2011) | T. Takahashi et al. (2012) |
| Ghaddar, Tamzalit, and Assaf (2011) | Tang, Jiang, Sun, Zhang, and Dong (2010) |
| Ghaddar, Tamzalit, Assaf, and Bitar (2012) | Terlecki, Bati, Galindo-Legaria, and Zabback (2009) |
| Glott, Husmann, Sadeghi, and Schunter (2011) | Truyen et al. (2012) |
| Guo et al. (2007) | C.-H. Tsai et al. (2007) |
| Guo et al. (2011) | W.-T. Tsai, Shao, Sun, and Elston (2010) |
| Haller (2011) | W.-T. Tsai, Sun, Shao, and Qi (2010) |
| Harris and Ahmed (2011) | W.-T. Tsai, Shao, and Elston (2010) |
| S. He, Guo, and Guo (2011) | W.-T. Tsai, Huang, and Shao (2011) |
| S. He et al. (2012) | W.-T. Tsai, Li, Bai, and Elston (2011) |
| Q. He, Han, Yang, Grundy, and Jin (2012) | W.-T. Tsai, Huang, Bai, and Gao (2012) |
| Hudli, Shivaradhya, and Hudli (2009) | Walraven et al. (2011) |
| Hui et al. (2009) | Z. Wang et al. (2008) |
| Jegadeesan and Balasubramaniam (2009) | D. Wang, Zhang, Zhang, and Liu (2009) |
| Jing and Zhang (2010) | M. Wang, Bandara, and Pahl (2010) |
| Ju, Sengupta, and Roychoudhury (2012) | H. Wang and Zheng (2010) |
| Kabbedijk and Jansen (2011) | R. Wang, Zhang, Liu, Wu, and Meng (2011) |
| Kang, Kang, and Hur (2011) | W. Wang et al. (2012) |
| Kapuruge, Colman, and Han (2011b) | Weiping (2009) |
| Kapuruge, Colman, and Han (2011a) | Weissman and Bobrowski (2009) |
| Kong, Li, and Zheng (2010) | Wood and Anderson (2011) |
| Koziolek (2011) | Xuxu, Qingzhong, and Lanju (2010) |
| Krebs, Momm, and Kounev (2012) | Yaish, Goyal, and Feuerlicht (2011) |
| Krebs, Momm, and Konev (2012) | E. Yang, Zhang, Wu, Liu, and Liu (2011) |
| Kurmus et al. (2011) | D. Yu et al. (2011) |
| Kwok, Nguyen, and Lam (2008) | H. Yu and Wang (2011) |
| Kwok and Mohindra (2008) | Yuanyuan, Hong, Bingfei, and Lei (2009) |
| Kwok, Laredo, and Maradugu (2008) | Y. Zhang et al. (2010) |
| Lang, Shankar, Patel, and Kalhan (2012) | K. Zhang, Li, and Shi (2011) |
| Lee and Choi (2012) | F. Zhang, Chen, Chen, and Zang (2011) |
| X. Li, Liu, Li, and Chen (2008) | Y. C. Zhou et al. (2010) |
| W. Li, Zhang, Wu, and Wu (2010) | Y. Zhou, Wang, Wang, and Wang (2011) |
| J. Li et al. (2012) | H. Zhou, Wen, and Yu (2011) |

# B Concept Matrices

Table 21: Concept Matrix - Multi-Tenancy Level Citations

| | Aulbach et al. (2008) | Chong et al. (2006) | Jacobs and Aulbach (2007) | Guo et al. (2007) | Kwok, Nguyen, and Lam (2008) | Kwok and Mohindra (2008) | Natis and Knipp (2008) | Osipov et al. (2009) | Reinwald (2010) | Taylor and Guo (2007) | Waidner (2009) | Z. Wang et al. (2008) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Azeez et al. (2010) | | × | | × | | | | | | | | |
| Barker et al. (2012) | × | | | | | | | | × | | | |
| Bezemer and Zaidman (2010) | | × | | | × | | | | | | | |
| Bobrowski (2011) | | | | | | | × | | | | | |
| Das et al. (2011) | | | | × | | | | | | | | |
| Elmore et al. (2011) | | | | × | | | | | | | | |
| Espadas et al. (2011) | | | × | | × | | | | | | | |
| Foping et al. (2009) | | | | × | | | | | | | | |
| Gao et al. (2011) | | | | | | | | | | | | × |
| Glott et al. (2011) | | | | | | | | | | | × | |
| Guo et al. (2011) | | | | | | | | | | × | | |
| Harris and Ahmed (2011) | | × | | | × | | | | | | | |
| Kong et al. (2010) | | × | | | | | | | | | | |
| Krebs, Momm, and Kounev (2012) | | × | × | | | | | × | | | | × |
| W. Li et al. (2010) | | | | | | | | | | | | × |
| Lin et al. (2009) | | | × | | | | | | | | | |
| K. Ma et al. (2012) | | | | × | | | | | | | | |
| Pathirage et al. (2011) | | × | | × | | | | | | | | |
| Plattner et al. (2011) | | | | × | | | | | | | | |
| Schiller et al. (2011) | | | × | × | | | | × | | | | |
| Tang et al. (2010) | | | | | | × | | | | | | |
| Yaish et al. (2011) | | | | | × | | | | | | | |
| D. Yu et al. (2011) | | | | | | | | | | | | × |
| Y. Zhou et al. (2011) | | × | | | | | | | | | | |

Table 22: Concept Matrix - Multi-Tenancy Levels

| | Hardware | Virtual Machine | Operating System | Middleware | Application Server | Database Server | Application Instance | Database | Schema |
|---|---|---|---|---|---|---|---|---|---|
| Aghera et al. (2012) | | | | | | × | | × | |
| Aulbach et al. (2008) | | | × | | × | × | | × | × |
| Azeez et al. (2010) | × | | × | × | | | × | | |
| Bezemer and Zaidman (2010) | × | | | | | | × | | |
| Cai et al. (2010) | × | | × | × | | × | × | × | × |
| Chong et al. (2006) | | | | | | × | | × | × |
| Domingo et al. (2010) | | | | | | × | | | × |
| Guo et al. (2007) | × | | | × | | | × | | |
| Guo et al. (2011) | × | | × | × | | | × | | |
| Hui et al. (2009) | | | | | | × | | × | × |
| Kabbedijk and Jansen (2011) | | | | | | | × | × | |
| Kwok, Nguyen, and Lam (2008) | | × | × | × | | × | × | × | × |
| Kwok and Mohindra (2008) | | × | × | × | | | × | | |
| Kurmus et al. (2011) | | × | × | | × | | | | |
| Lang et al. (2012) | × | | | | | × | | × | × |
| X. Li et al. (2008) | × | | × | × | | | × | | |
| Lin et al. (2009) | | | × | | | | | | |
| Natis and Knipp (2008) | × | | × | | × | × | × | | |
| Osipov et al. (2009) | × | × | × | × | | × | × | × | × |
| Pippal et al. (2011) | | | | | | × | | × | × |
| Reinwald (2010) | × | × | × | | | × | | × | × |
| Rodero-Merino et al. (2012) | | × | × | | | | | | |
| X. Sun et al. (2012) | × | | | | × | × | × | × | × |
| Taylor and Guo (2007) | | | | | | × | | × | × |
| Truyen et al. (2012) | × | × | | × | | | × | | |
| Waidner (2009) | × | × | × | × | | × | × | × | × |
| Walraven et al. (2011) | × | × | | × | | | × | | |
| Z. Wang et al. (2008) | | | | | | × | | × | × |
| W. Wang et al. (2012) | | | × | × | | | × | | |

Table 23: Concept List - Decision Criteria

| Article | Criteria |
|---|---|
| Aghera et al. (2012) | backup, database access, monitoring, configurability, extensibility, database privacy |
| Almorsy et al. (2012) | availability, customization, configurability, resource utilization efficiency, security, identity management, authorization, cryptography, authentication |
| Almutairi et al. (2012) | security, access control, authorization |
| Aulbach et al. (2008) | contention for shared resources, extensibility, security, number of tenants, budget of tenant, complexity of application |
| Aulbach et al. (2009) | performance, response time, extending base schema, evolution of base schema, economy of scale, scalability |
| Aulbach et al. (2011) | development, complexities in app. development, support for master data, variability, extending base schema, evolution of base schema, resource utilization efficiency, operating cost |
| Azeez et al. (2010) | performance, throughput, response time, administration, scalability, security, number of tenants |
| Barker et al. (2012) | system uptime, flexibility in database migration, query latency, throughput, development, overhead, memory overhead |
| Bezemer et al. (2010) | system downtime, maintenance, upfront app. reengineering costs, deployment, fault isolation, configurability, resource utilization efficiency, scalability, security, authentication |
| Bezemer and Zaidman (2010) | system downtime, performance, maintenance, deployment, data aggregation opportunities, update, customization, configurability, resource utilization efficiency, memory overhead, scalability, security, number of tenants |
| Bobrowski (2011) | backup, data integration, recovery, performance, complexities in app. development, re-education process for developers, monitoring, software patching, overhead, scalability |
| Brassil (2010) | performance, management, complexities in app. development, dynamic reconfiguration, security, privacy, robustness to failure |
| Das et al. (2011) | performance, query latency, extending base schema, resource utilization efficiency, scalability, elastic scaling, footprint of tenant |
| Elmore et al. (2011) | on-demand tenant migration, flexibility, resource utilization efficiency, overhead |
| Espadas et al. (2011) | customization, scalability, number of tenants |
| Foping et al. (2009) | implementation challenges, customization, configurability, extensibility, scalability, security, operating cost |
| Gao et al. (2011) | availability, transactions per second, development, flexibility, customization, diverse SLA, scalability, security, operating cost |
| Ghaddar et al. (2011) | development, complexities in app. development, re-education process for developers, administration, variability |
| Ghaddar et al. (2012) | maintenance, deployment, variability |
| Guo et al. (2007) | availability, backup, restore, performance, management, administration, customization, configurability, scalability, security, access control, authentication, information protection, number of tenants, budget of tenant, quality of service, business logic monitoring, service integration, service subscription, upgrade |
| Guo et al. (2011) | availability, backup, restore, performance, maintenance, management, development, update, customization, configurability, scalability, security, access control, information protection, QoS isolation |
| Haller (2011) | tenant import, export, update |
| Harris and Ahmed (2011) | availability, system downtime, performance, maintenance, upfront app. reengineering costs, versioning, update, configurability, resource utilization efficiency, overhead, scalability, security, authentication, number of tenants |
| S. He et al. (2012) | availability, throughput, response time, number of end-users |
| Hudli et al. (2009) | development, support, customization, infrastructure cost, scalability, security, operating cost |
| Hui et al. (2009) | maintenance, scalability, security |

Continued on next page

**Table 23 – continued from previous page**

| Article | Criteria |
|---|---|
| Jegadeesan and Balasubramaniam (2009) | configurability |
| Ju et al. (2012) | performance, development, flexibility, security |
| Kabbedijk and Jansen (2011) | performance, maintenance, variability, number of tenants |
| Kang et al. (2011) | maintenance, configurability, scalability |
| Kapuruge et al. (2011a) | variability |
| Kong et al. (2010) | implementation challenges, customization, configurability, extensibility, scalability, security |
| Koziolek (2011) | maintenance, customization, resource utilization efficiency, scalability, elasticity |
| Krebs, Momm, and Konev (2012) | performance, customization, configurability, QoS differentiation, overhead, operating cost |
| Kurmus et al. (2011) | performance, development, scalability, security |
| Kwok, Nguyen, and Lam (2008) | customization, security, authorization, authentication |
| Kwok and Mohindra (2008) | maintenance, deployment, update, flexibility, infrastructure cost, software license fees, security, number of tenants, support staff, provision |
| Lang et al. (2012) | performance, management, security, operating cost |
| X. Li et al. (2008) | performance, management, resource utilization efficiency |
| W. Li et al. (2010) | maintenance, development, complexities in app. development, administration, customization, configurability, diverse SLA, scalability, security, authorization |
| Lin et al. (2009) | performance, throughput, response time, management, dynamic resources, customization, resource utilization efficiency, overhead, security, number of tenants, regulation |
| K. Ma et al. (2012) | backup, restore, maintenance, development, security, number of tenants |
| Mietzner et al. (2008) | performance, configurability, economy of scale, regulation |
| Mietzner, Metzger, et al. (2009) | flexibility, economy of scale, data privacy |
| Mietzner, Unger, et al. (2009) | performance, configurability |
| Momm and Theilmann (2011) | risk of overload situations, customization, resource utilization efficiency |
| Pathirage et al. (2011) | resource utilization efficiency |
| Pippal et al. (2011) | performance, development, flexibility, scalability, authorization, authentication |
| Plattner et al. (2011) | flexibility in database migration, administrative operations in bulk, contention for shared resources, ability to realize query optimization, economy of scale, security |
| Rimal and El-Refaey (2010) | availability, business continuity, data availability, real-time replication, backup, fail-over and dynamic election, partial data and config. recovery, recovery, maintenance, customization, extensibility, economy of scale, scalability, security, data access protection, number of end-users, regulation |
| Rodero-Merino et al. (2012) | resource utilization efficiency, security |
| Ruehl et al. (2012) | performance, flexibility, customization, security, privacy |
| Schiller et al. (2011) | backup, recovery, data dictionary lookup times, complexities in app. development, statistic, extending base schema, resource utilization efficiency, overhead, scalability, security, main memory per tenants, number of tenants, number of end-users, footprint of tenant |
| Schroeter et al. (2012) | configurability, resource utilization efficiency, number of tenants, operating cost |
| Sengupta and Roychoudhury (2011) | performance, maintenance, implementation challenges, flexibility, customization, configurability, overhead, security |
| W. Sun et al. (2008) | customization, configurability |
| X. Sun et al. (2012) | performance, throughput, resource utilization efficiency, overhead, security, number of tenants |

Continued on next page

**Table 23 – continued from previous page**

| Article | Criteria |
|---|---|
| T. Takahashi et al. (2012) | maintenance, update, security, authorization, secure data storing, authentication, operating cost |
| Tang et al. (2010) | performance, security, number of tenants |
| Terlecki et al. (2009) | extensibility, budget of tenant |
| Truyen et al. (2012) | maintenance, complexities in app. development, customization, resource utilization efficiency, scalability, operating cost |
| C.-H. Tsai et al. (2007) | correctness of results, response time, overhead, scalability, number of tenants, operating cost |
| W.-T. Tsai, Shao, Sun, and Elston (2010) | availability, performance, management, development, customization, overhead, scalability, security |
| W.-T. Tsai et al. (2012) | recovery, fault tolerance, database access, scalability, automated migration, number of end-users |
| Walraven et al. (2011) | maintenance, upfront app. reengineering costs, flexibility, infrastructure cost, number of tenants |
| Z. Wang et al. (2008) | backup, restore, transactions per second, management, development, lifecycle management, monitoring, customization, infrastructure cost, scalability, security, number of tenants |
| D. Wang et al. (2009) | maintenance, development, resource utilization efficiency, number of tenants |
| H. Wang and Zheng (2010) | configurability |
| W. Wang et al. (2012) | performance, infrastructure cost, overhead, scalability |
| Weiping (2009) | management, complexities in app. development, deployment |
| Weissman and Bobrowski (2009) | amount of code bases, size of administrative staff, data aggregation opportunities, update, economy of scale, resource utilization efficiency, budget of tenant |
| Wood and Anderson (2011) | performance, customization, economy of scale, scalability, security, privacy, information protection, legislation |
| Yaish et al. (2011) | performance, management, development, complexities in app. development, customization, overhead |
| E. Yang et al. (2011) | availability, throughput, response time, overhead, number of end-users, reliability |
| D. Yu et al. (2011) | performance, customization, resource utilization efficiency, security, number of end-users |
| H. Yu and Wang (2011) | backup, restore, extensibility, security |
| Yuanyuan et al. (2009) | configurability, scalability, security |
| Y. Zhang et al. (2010) | number of tenants |
| K. Zhang et al. (2011) | customization, data privacy |
| Y. Zhou et al. (2011) | backup, restore, performance, maintenance, statistic, customization, scalability, security |
| H. Zhou et al. (2011) | usage metering, billing |

Table 24: Frequency List of Decision Criteria

| Criterion | $f$ | Criterion | $f$ |
|---|---|---|---|
| security | 39 | ability to realize query optimization | 1 |
| scalability | 31 | administrative operations in bulk | 1 |
| performance | 29 | amount of code bases | 1 |
| customization | 27 | automated migration | 1 |
| configurability | 20 | billing | 1 |
| resource utilization efficiency | 20 | business continuity | 1 |
| maintenance | 19 | business logic monitoring | 1 |
| number of tenants | 19 | complexity of application | 1 |
| development | 15 | correctness of results | 1 |
| overhead | 14 | cryptography | 1 |
| backup | 10 | data access protection | 1 |
| management | 10 | data availability | 1 |
| operating cost | 10 | data dictionary lookup times | 1 |
| availability | 9 | data integration | 1 |
| complexities in app. development | 9 | database privacy | 1 |
| flexibility | 9 | delete | 1 |
| authentication | 7 | dynamic reconfiguration | 1 |
| economy of scale | 7 | dynamic resources | 1 |
| extensibility | 7 | elastic scaling | 1 |
| update | 7 | elasticity | 1 |
| authorization | 6 | export | 1 |
| number of end-users | 6 | fail-over and dynamic election | 1 |
| response time | 6 | fault isolation | 1 |
| restore | 6 | fault tolerance | 1 |
| throughput | 6 | identity management | 1 |
| deployment | 5 | legislation | 1 |
| infrastructure cost | 5 | lifecycle management | 1 |
| variability | 5 | main memory per tenants | 1 |
| administration | 4 | merge & split | 1 |
| budget of tenant | 4 | on-demand tenant migration | 1 |
| extending base schema | 4 | partial data and config. recovery | 1 |
| recovery | 4 | provision | 1 |
| access control | 3 | QoS differentiation | 1 |
| implementation challenges | 3 | QoS isolation | 1 |
| information protection | 3 | quality of service | 1 |
| monitoring | 3 | real-time replication | 1 |
| privacy | 3 | reliability | 1 |
| regulation | 3 | risk of overload situations | 1 |
| system downtime | 3 | robustness to failure | 1 |
| upfront app. reengineering costs | 3 | secure data storing | 1 |
| contention for shared resources | 2 | service integration | 1 |
| data aggregation opportunities | 2 | service subscription | 1 |
| data privacy | 2 | size of administrative staff | 1 |
| database access | 2 | software license fees | 1 |
| diverse SLA | 2 | software patching | 1 |
| evolution of base schema | 2 | support | 1 |
| flexibility in database migration | 2 | support for master data | 1 |
| footprint of tenant | 2 | support staff | 1 |
| memory overhead | 2 | system uptime | 1 |
| query latency | 2 | tenant import | 1 |
| re-education process for developers | 2 | upgrade | 1 |
| statistic | 2 | usage metering | 1 |
| transactions per second | 2 | versioning | 1 |

# C Decision Criteria Minimization Process

This section illustrates the steps taken to minimize the initially large set of criteria. A total of four steps can be identified, the same steps as those described in Table 12. In each activity a number of attributes are combined with one or two other attributes. An attribute is displayed as a box, within its name and frequency presented. Arrows show combinations between merges of criteria, the frequency of the resulting attribute is the sum of the frequencies of the criteria combined. Because the list of criteria is so large, the process is displayed in three parts.

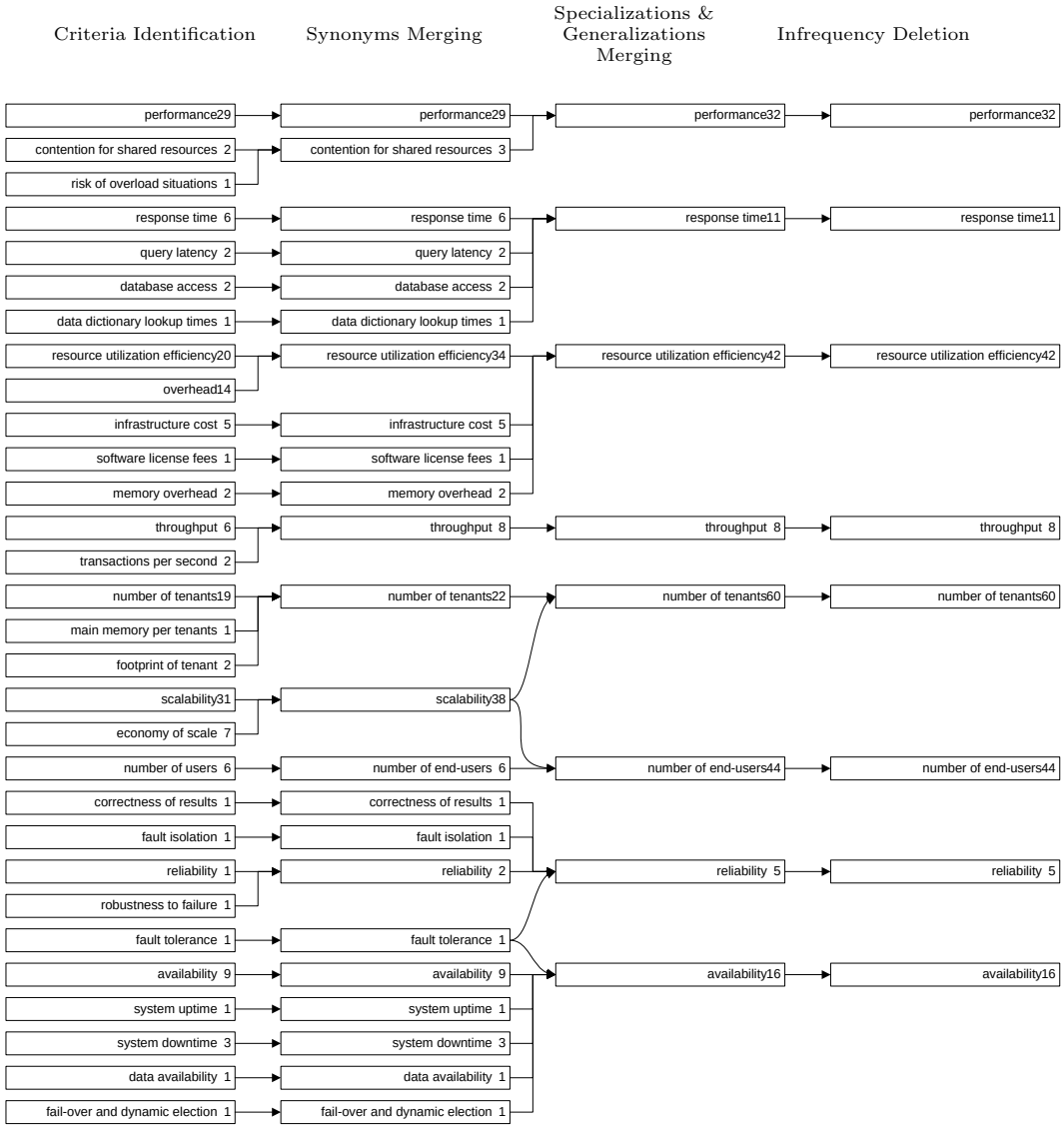



Figure 27: Decision Criteria Minimization Process: Part 1

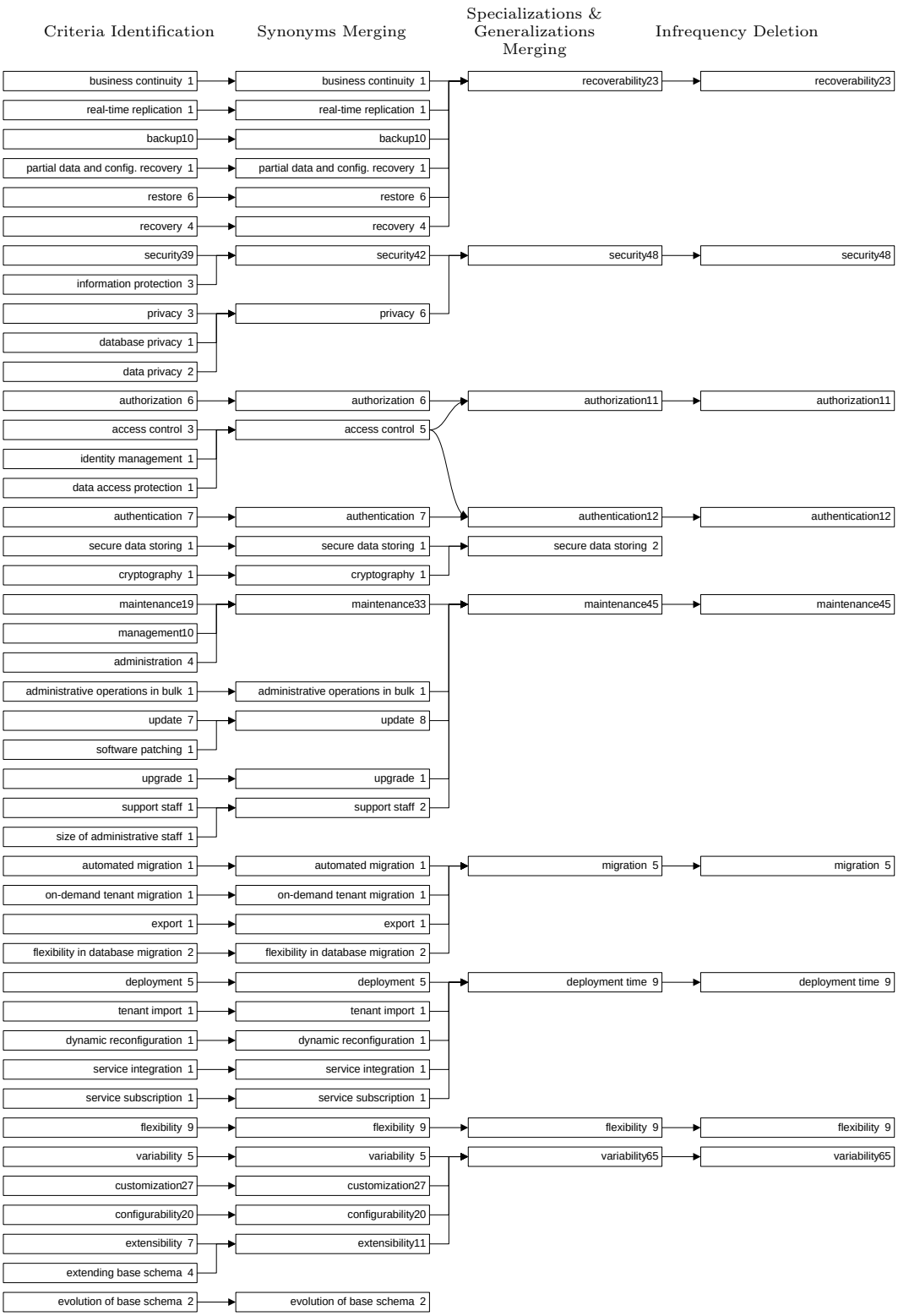



Figure 28: Decision Criteria Minimization Process: Part 2

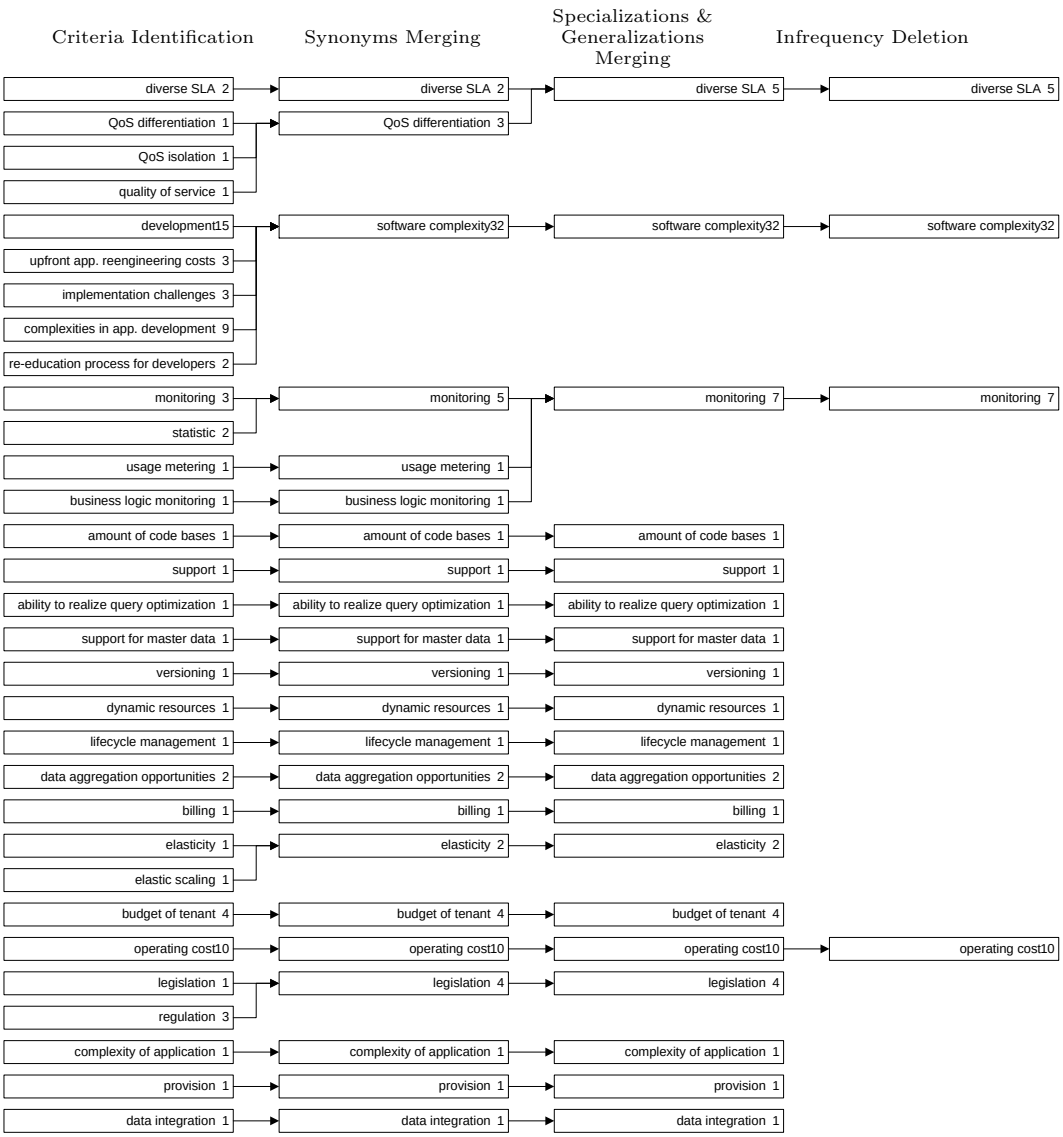



Figure 29: Decision Criteria Minimization Process: Part 3

## D Evaluation Questionnaire Template

# A decision model for multi-tenant architectures

In my research I will develop a decision model that supports choosing the most suitable multi-tenant architecture. For this, I would like you to evaluate the multi-tenant architectures and decision criteria identified from literature. The questionnaire will approximately take 30 minutes to complete. Your answers will be processed anonymously.

The questionnaire makes use of the following definitions:

**Application server**
: A computer program providing services to software applications

**Application instance**
: A copy of an executable of the program written to a computer's memory

**Database server**
: A computer program providing services to another computer program

**Database**
: A structured collection of data

**Database schema**
: A collection of tables

A service provider is hosting and developing its own software. The provider needs to make a decision what type of multi-tenant architecture it will use for its software system. These architectures differ on the extent of shared resources. At the **application tier** a provider has the following three options:

1. For each tenant a dedicated application server is running.
2. For multiple tenants a single application server is running, where for each tenant a dedicated application instance is running.
3. For multiple tenants a single application server is running, where for multiple tenants a single application instance is running.

At the **data tier** a provider has the following four options:

1. For each tenant a dedicated database server is running.
2. For multiple tenants a single database server is running, where for each tenant a dedicated database is running.
3. For multiple tenants a single database server is running with a single database, where for each tenant a dedicated schema exist.
4. For multiple tenants a single database server is running with a single database and a single schema.

Consequently, there are $3 \times 4 = 12$ possible architectures from which the service provider can choose.

# Multi-tenant Architectures

Name (optional): .....

Date: .....

Job: .....

Years of service: .....

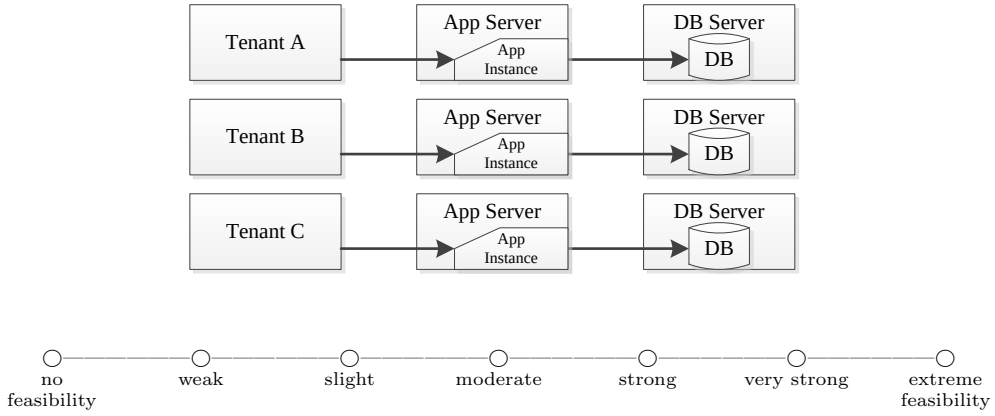
Product type: .....

Determine for each of the twelve multi-tenant architectures to what extent you think it represents a feasible multi-tenant architecture which can be used in a real system.

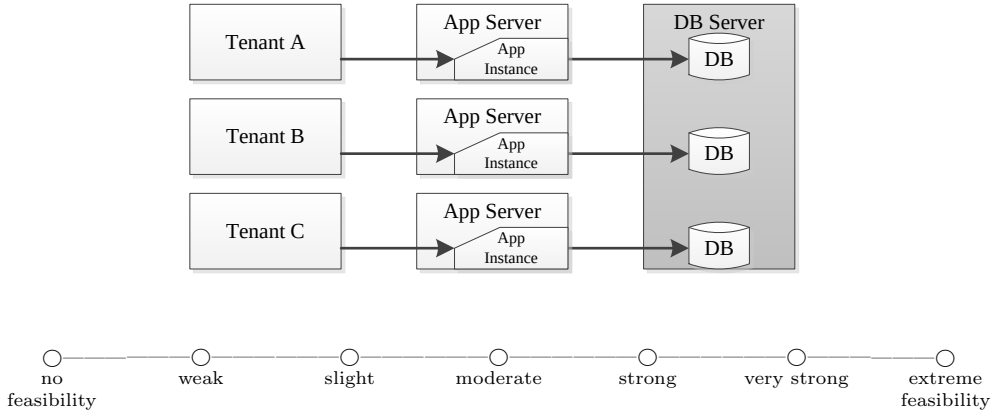
The following scale is used:

1. = no feasibility
2. = weak feasibility
3. = slight feasibility
4. = moderate feasibility
5. = strong feasibility
6. = very strong feasibility
7. = extreme feasibility

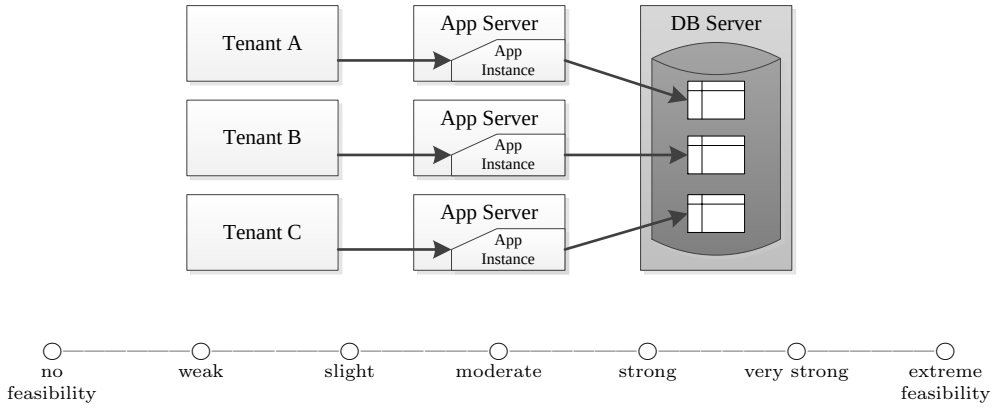
1. dedicated application server ⇔ dedicated database server, dedicated databases



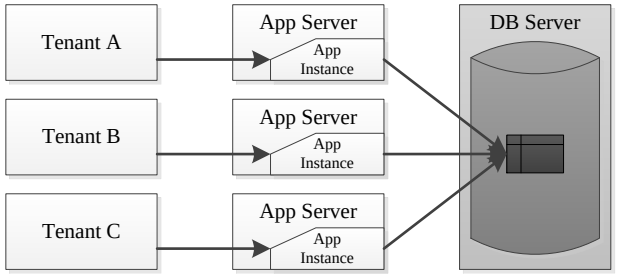

2. dedicated application server ⇔ **shared** database server, dedicated databases

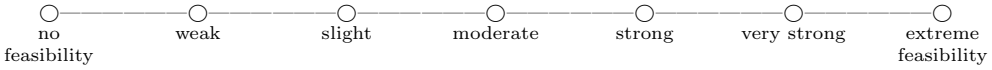



3. dedicated application server ⇔ **shared** database server, **shared** databases, dedicated schema's

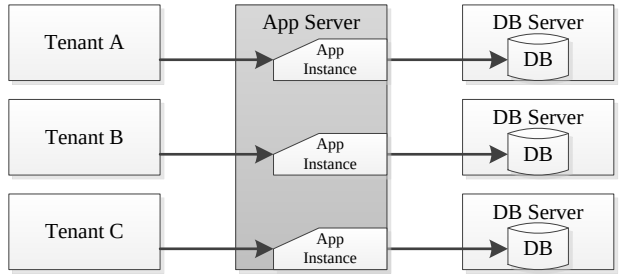


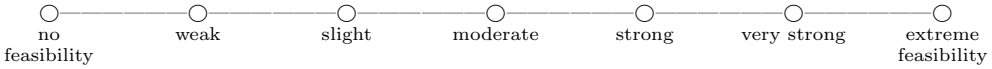
4. dedicated application server ⇔ **shared** database server, **shared** databases, **shared** schema's



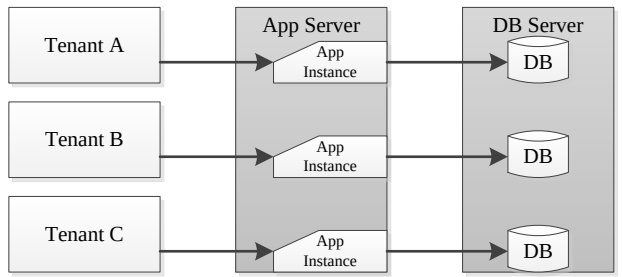



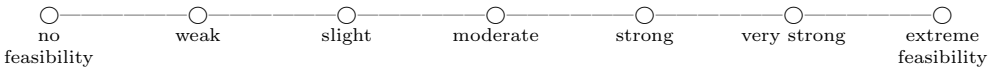

5. **shared** application server, dedicated application instance ⇔ dedicated database server, dedicated databases



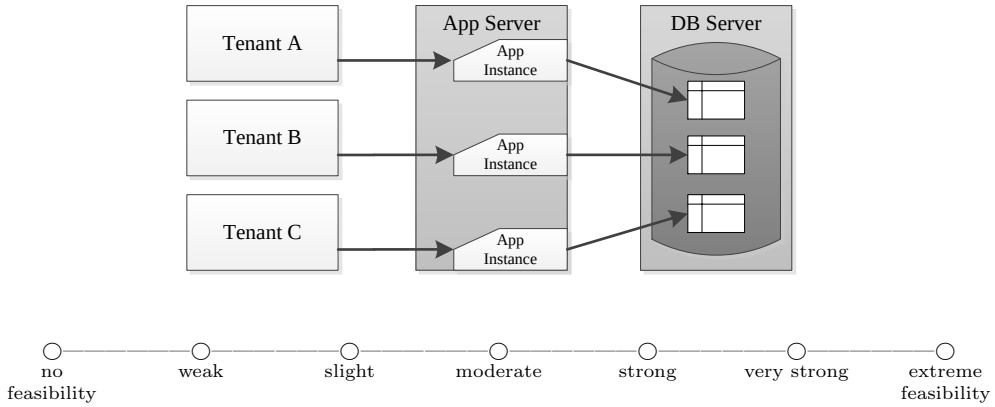




6. **shared** application server, dedicated application instance ⇔ **shared** database server, dedicated databases

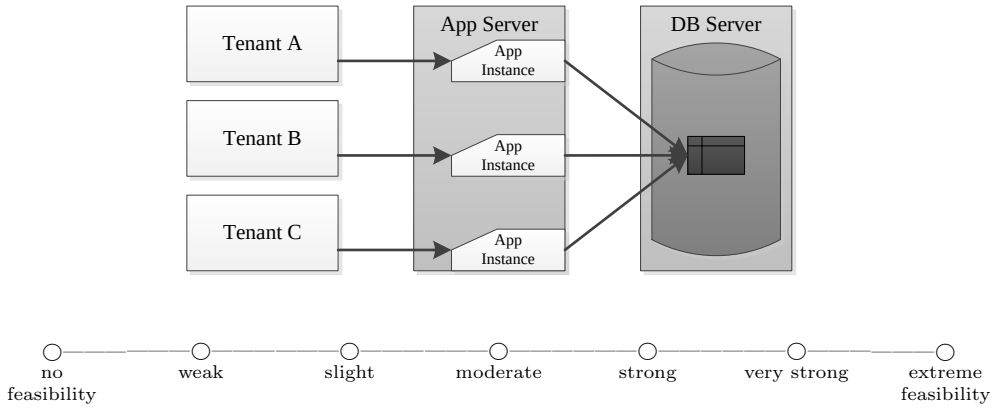


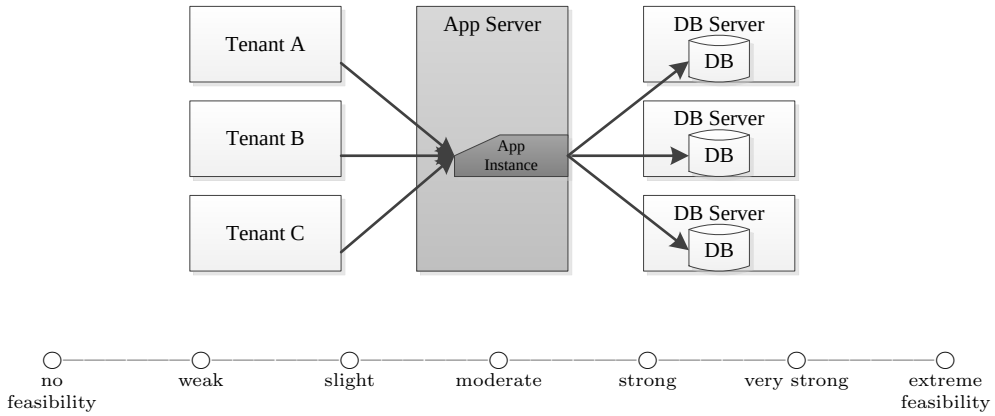
7. **shared** application server, dedicated application instance ⇔ **shared** database server, **shared** databases, dedicated schema's



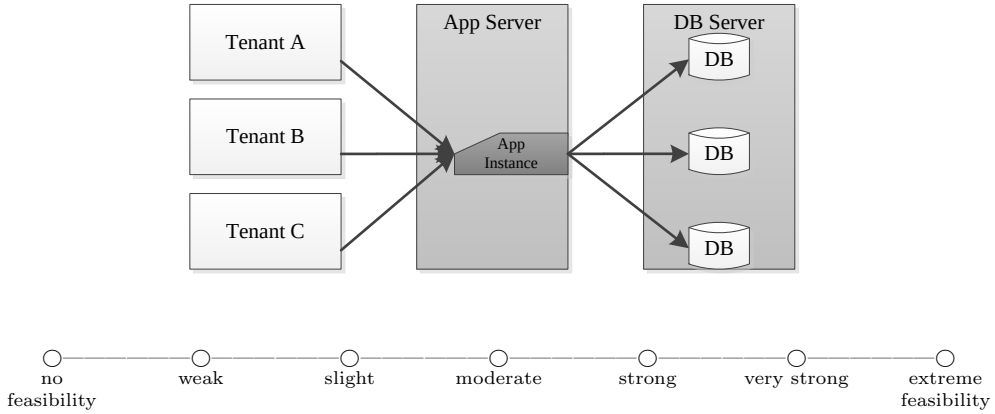

8. **shared** application server, dedicated application instance ⇔ **shared** database server, **shared** databases, **shared** schema's



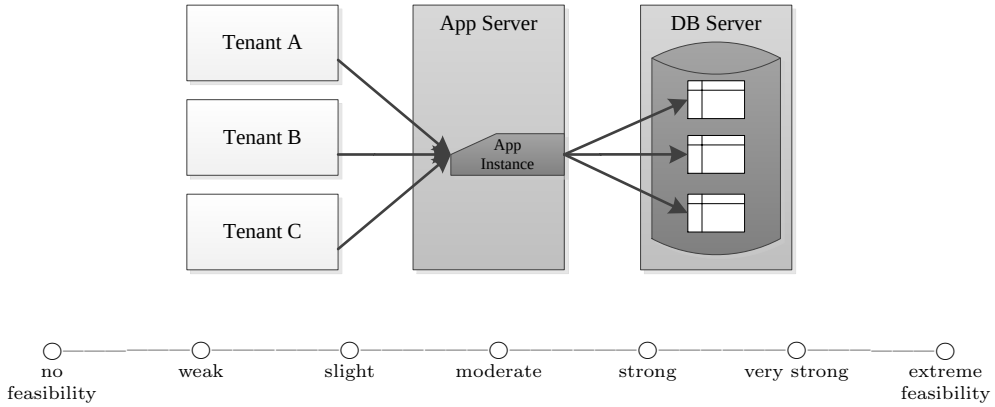

9. **shared** application server, **shared** application instance ⇔ **shared** database server, dedicated databases



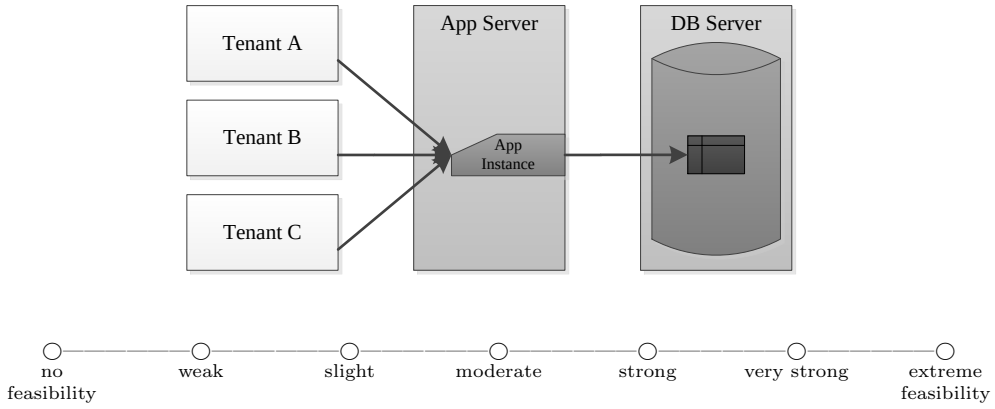
10. **shared** application server, **shared** application instance ⇔ **shared** database server, dedicated databases




11. **shared** application server, **shared** application instance ⇔ **shared** database server, **shared** databases, dedicated schema's




12. **shared** application server, **shared** application instance ⇔ **shared** database server, **shared** databases, **shared** schema's



# Decision Criteria

The multi-tenant architectures are evaluated on a set of decision criteria. These criteria represent attributes that discriminate among the different multi-tenant architectures, i.e. they reflect differences among the architectures. In addition, the criteria are based on goals the architectures should achieve.

What follows is a list of decision criteria. Each criterion is briefly described. Evaluate the criteria on each of the following points.

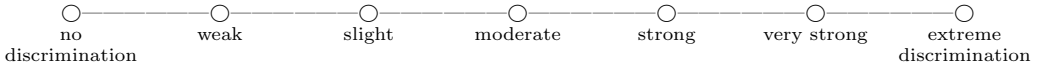
1. Define for each criterion to what extent you think the criterion **discriminates** among the different multi-tenant architectures. In other words, to what extent the criterion **defines a difference** among the multi-tenant architectures. This difference can refer to a difference in implementation complexity of the corresponding criterion too.
2. Define for each criterion to what extent you think the criterion is a **deciding factor** in the decision process for the service provider. In other words, to what extent is the criterion of **impact** and **relevant** for the service provider to evaluate the different multi-tenant architectures.

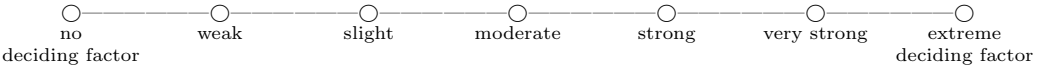
The following scale is used:

1. = no discrimination or deciding factor
2. = weak discrimination or deciding factor
3. = slight discrimination or deciding factor
4. = moderate discrimination or deciding factor
5. = strong discrimination or deciding factor
6. = very strong discrimination or deciding factor
7. = extreme discrimination or deciding factor

# 1 Performance Efficiency

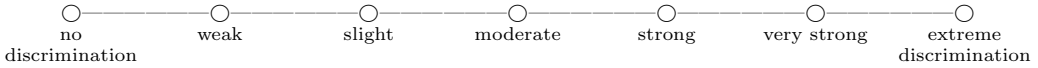
Performance relative to the amount of resources used under stated conditions.

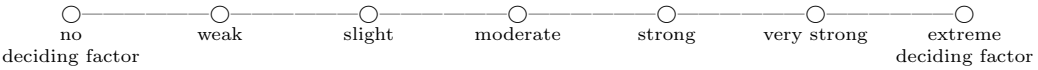



## 1.1 Time Behavior

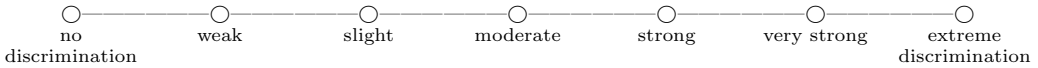
Degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements.

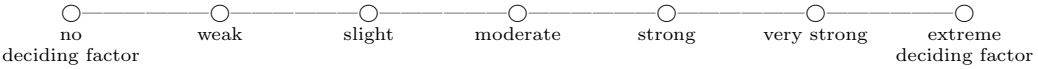



## 1.2 Resource Utilization

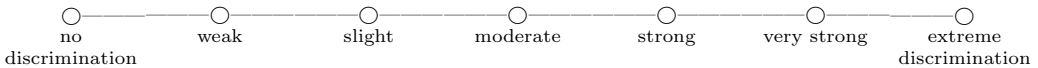
Degree to which the amounts and types of resources used by a product or system when performing its functions meet requirements.

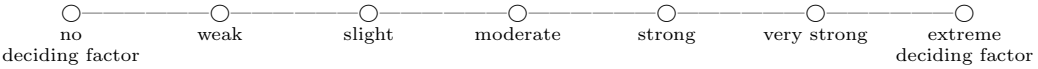



## 1.3 Capacity

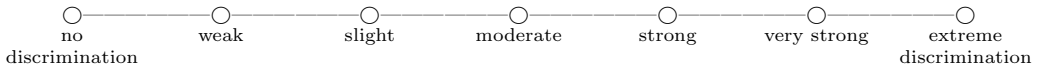
Degree to which the maximum limits of a product or system parameter meet requirements. Parameters can include the number of items that can be stored, the number of concurrent users, the communication bandwidth, throughput of transactions, and size of database.

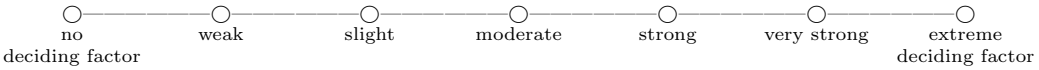


### 1.3.1 Throughput

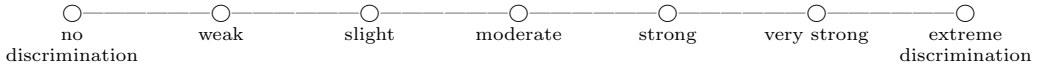
The average rate of successful message delivery over a communication channel, measured in bits per second.

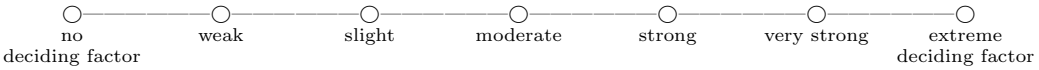



### 1.3.2 Number of Tenants

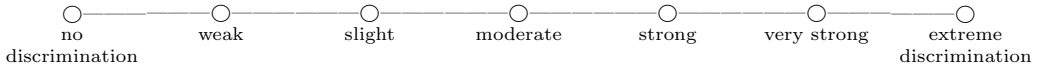
The extent to which the system can be scaled so it can be offered to multiple tenants.

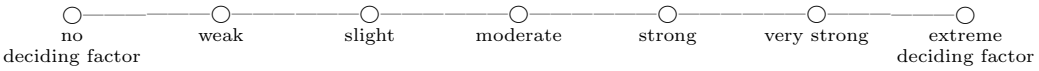



### 1.3.3 Number of End-Users

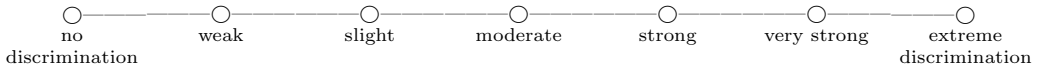
The extent to which the system can be scaled so it can be offered to multiple end-users.

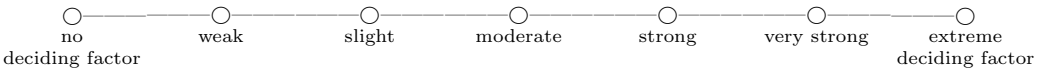



# 2 Reliability

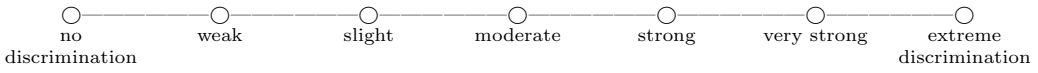
Degree to which a system, product or component performs specified functions under specified conditions for a specified period of time.

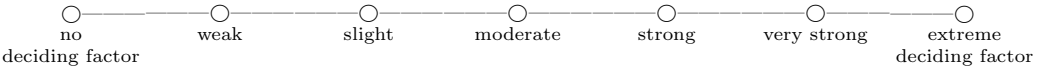


## 2.1 Availability

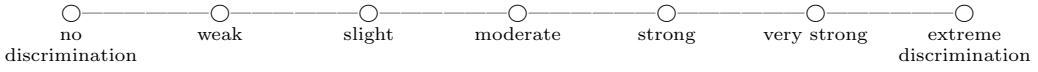
Degree to which a system, product or component is operational and accessible when required for use.

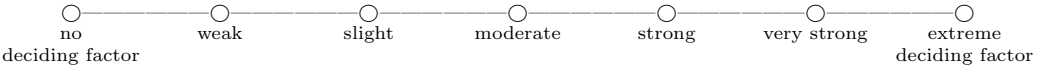



## 2.2 Recoverability

Degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system.

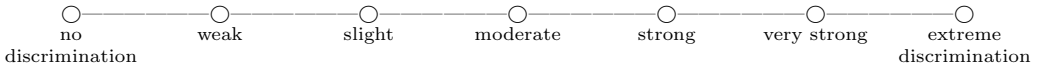


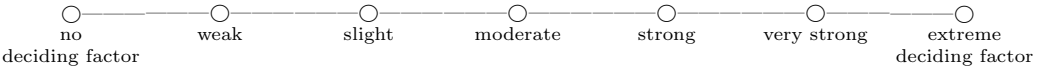




# 3 Security

Degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization.

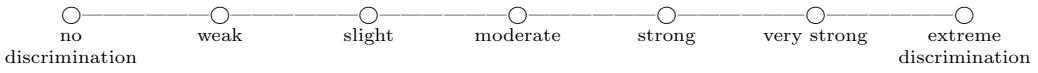
NB Survivability (the degree to which a product or system continues to fulfill its mission by providing essential services in a timely manner in spite of the presence of attacks) is covered by recoverability (2.2).

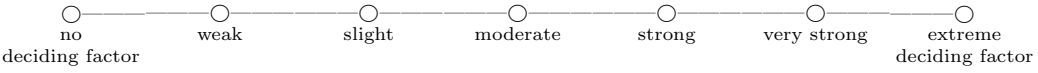


## 3.1 Confidentiality

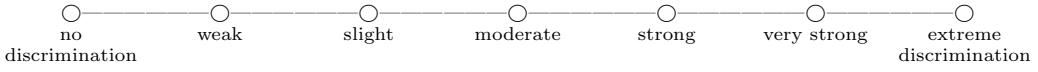
Degree to which a product or system ensures that data are accessible only to those authorized to have access.

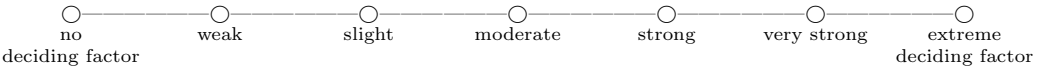



## 3.2 Integrity

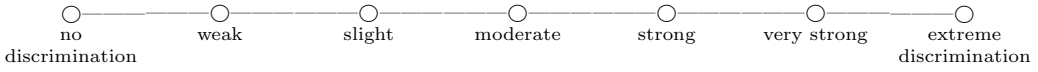
Degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data.

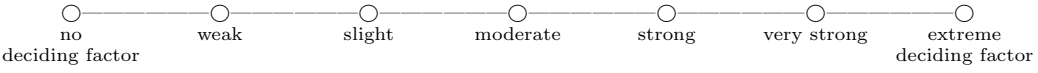



## 3.3 Authenticity

Degree to which the identity of a subject or resource can be proved to be the one claimed.

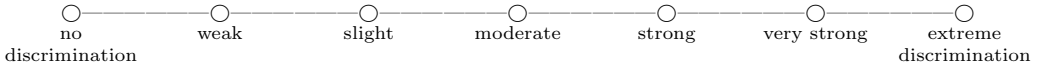


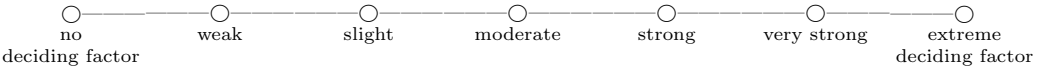



# 4 Maintainability

Degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers. Modifications can include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specifications. Modifications include those carried out by specialized support staff, and those carried out by business or operational staff, or end users.

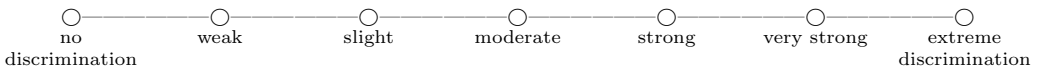
NB Maintainability includes installation of updates and upgrades.

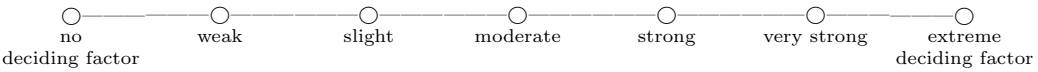



# 5 Portability

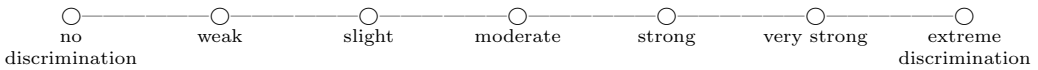
Degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another.

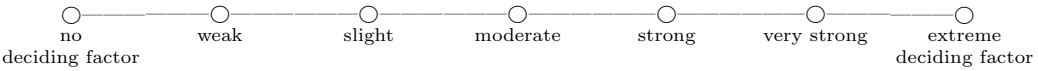



# 6 Deployment Time

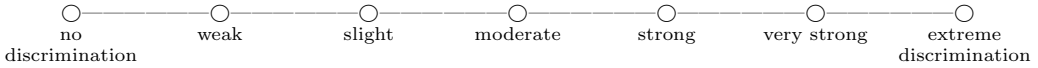
Degree of effectiveness and efficiency to which the software product can be made available for use.

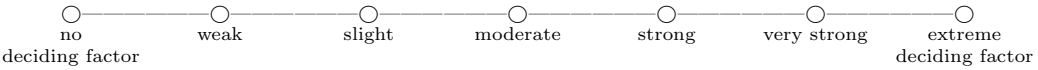


# 7 Flexibility

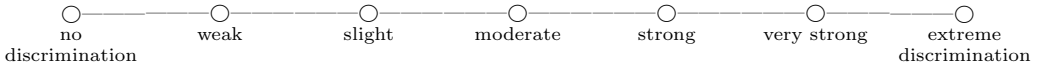
Degree to which the system can support different functional and non-functional requirements of different tenants.

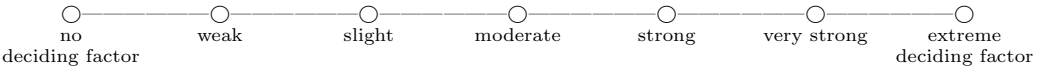



## 7.1 Variability

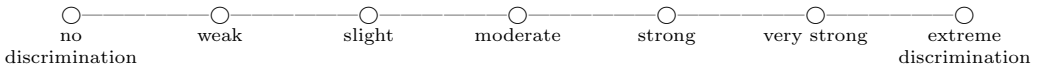
Degree to which the system can support customized solutions and tenant-dependent configurations, extension and evolution.

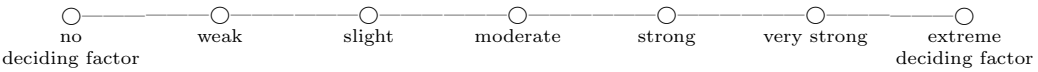



## 7.2 Diverse Service Level Agreement

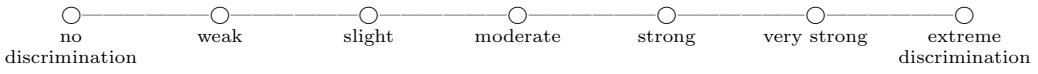
Degree to which the system can support a variety of service level agreements to tenants.

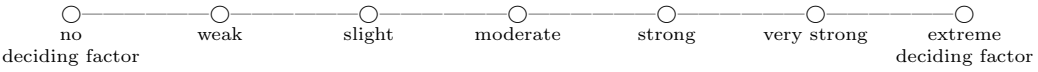


# 8 Software Complexity

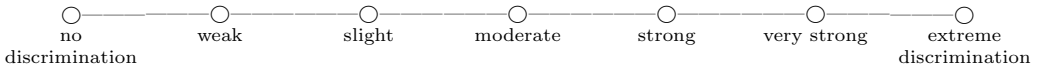
Degree of the software complexity of the software product if it is developed and implemented in a multi-tenant architecture.

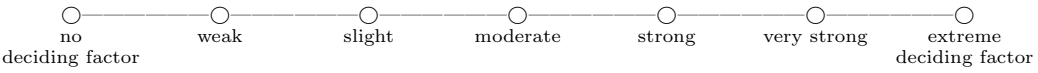



# 9 Monitoring

Degree of ease to which monitoring and controlling tasks can be carried out in the system. Tasks include controlling server availability, user activity, capacity and performance.



## E Rating Questionnaire Template

Note that if an expert defined a multi-tenant architecture as not or weakly feasible in the previous questionnaire, this multi-tenant architecture is omitted from this questionnaire and therefore not ranked by that expert.

# Ratings of multi-tenant architectures with respect to decision criteria

This research is focused on the development of a decision model that supports service providers in choosing the best suitable multi-tenant architecture. Earlier, a set of multi-tenant architectures is identified. This set exists of options service providers can select and includes all generic multi-tenant architectures. They differ in the extent to which resources in the application tier and the data tier are shared among tenants.

The architectures are evaluated against a set of decision criteria. Each criterion defines a discrimination among the architectures and is a deciding factor in the decision of the service provider. These criteria en the multi-tenant architectures are evaluated earlier, inter alia, by you.

To complete the development of the decision model it's of importance to actually evaluate the architectures against the criteria. This evaluation represent the performance of an architecture with respect to a criterion. You are asked at each criterion to rate the performance of each architecture with respect to that criterion. This performance score is expressed in the following terms:

1. = poor
2. = below average
3. = average
4. = good
5. = excellent

# Time Behavior

Degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements.


# Resource Utilization

Degree to which the amounts and types of resources used by a product or system when performing its functions meet requirements.

# Throughput

The average rate of successful message delivery over a communication channel, measured in bits per second.


# Number of tenants

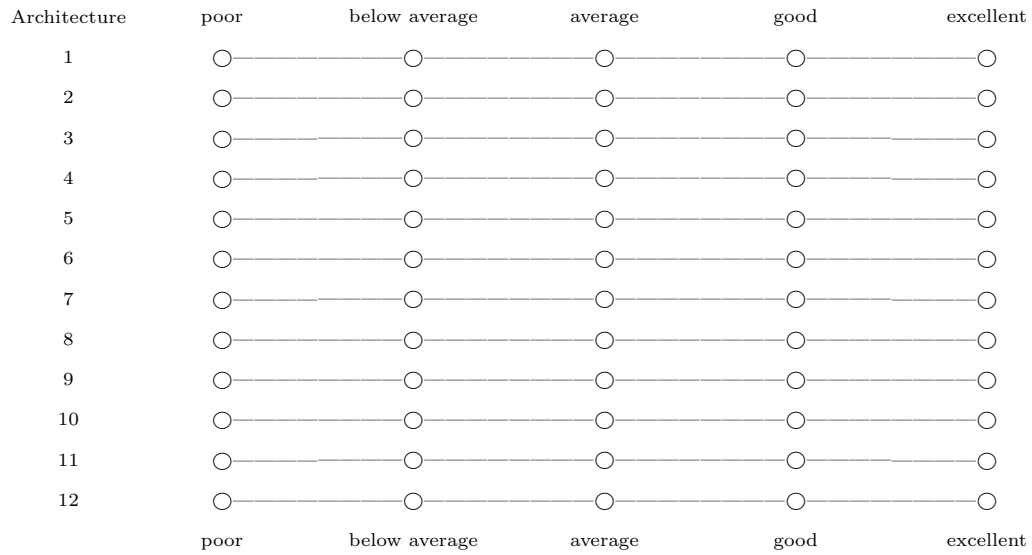
The extent to which the system can be scaled so it can be offered to multiple tenants.

# Number of end-users

The extent to which the system can be scaled so it can be offered to multiple end-users.


# Availability

Degree to which a system, product or component is operational and accessible when required for use.

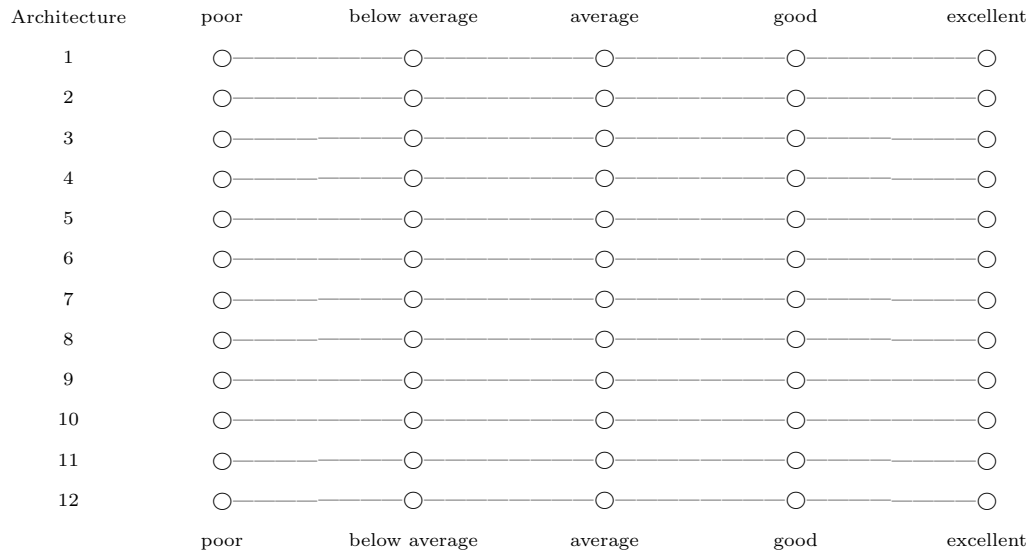


# Recoverability

Degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system.

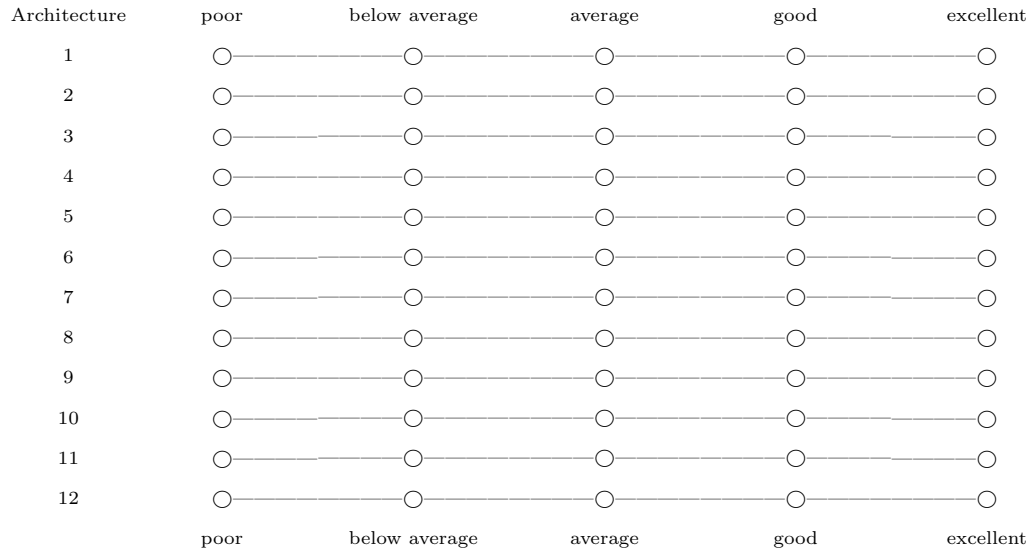

# Confidentiality

Degree to which a product or system ensures that data are accessible only to those authorized to have access.



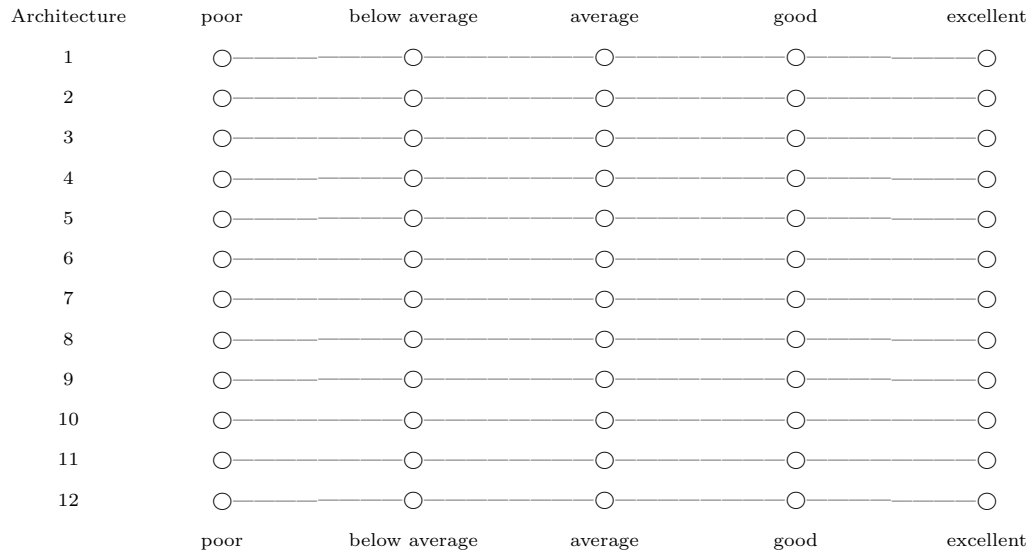
# Integrity

Degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data.




# Authenticity

Degree to which the identity of a subject or resource can be proved to be the one claimed.

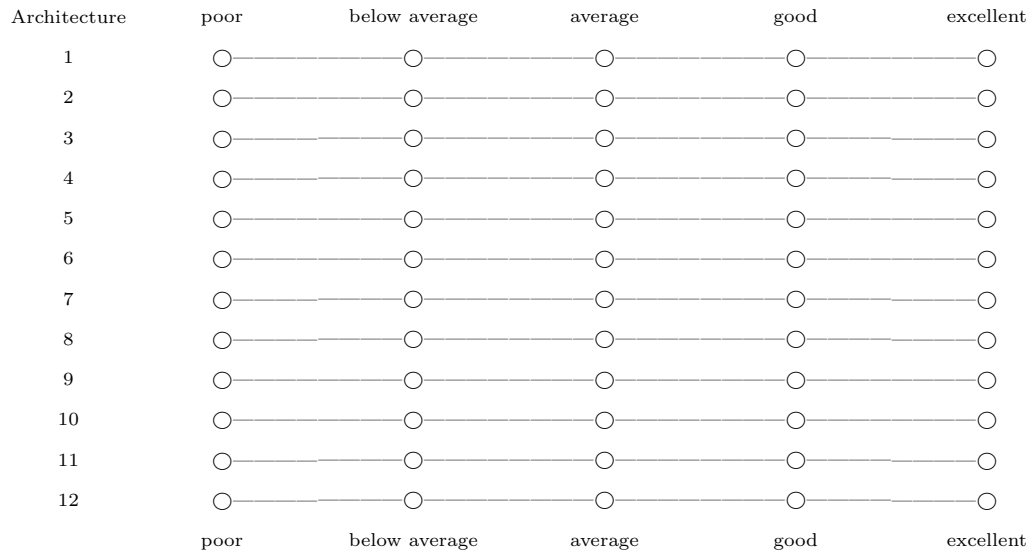


# Portability

Degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another.

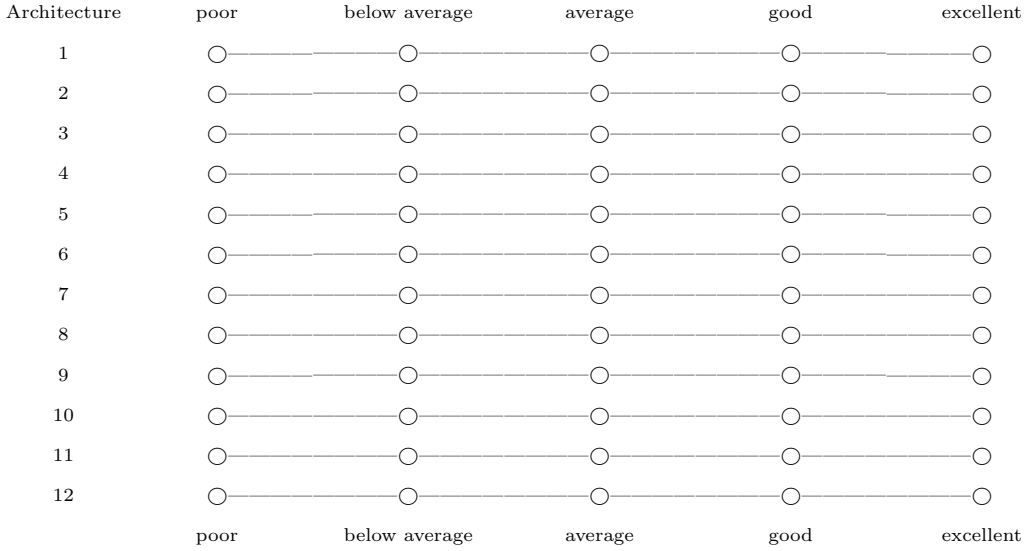

# Deployment time

Degree of effectiveness and efficiency to which the software product can be made available for use.



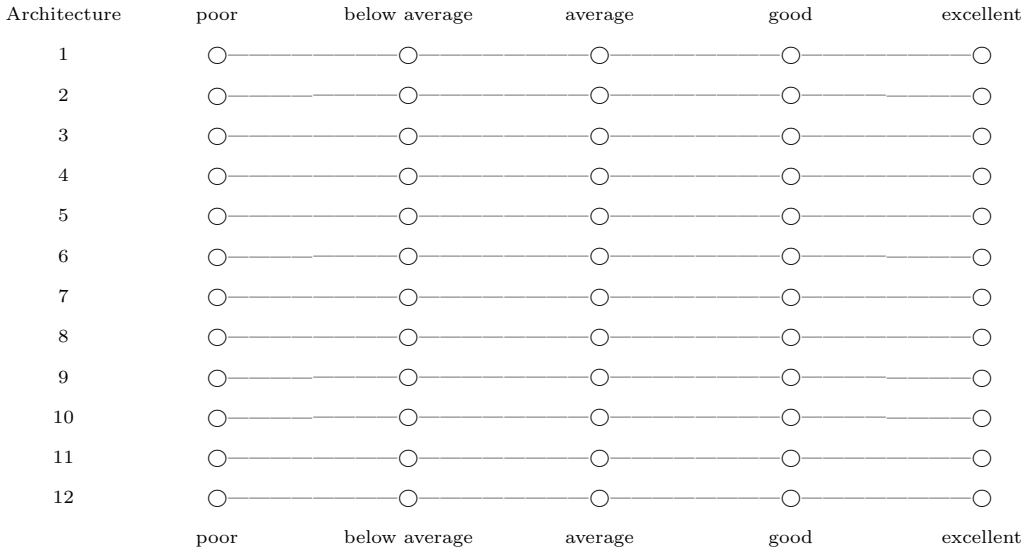
# Variability

Degree to which the system can support customized solutions and tenant-dependent configurations, extension and evolution.



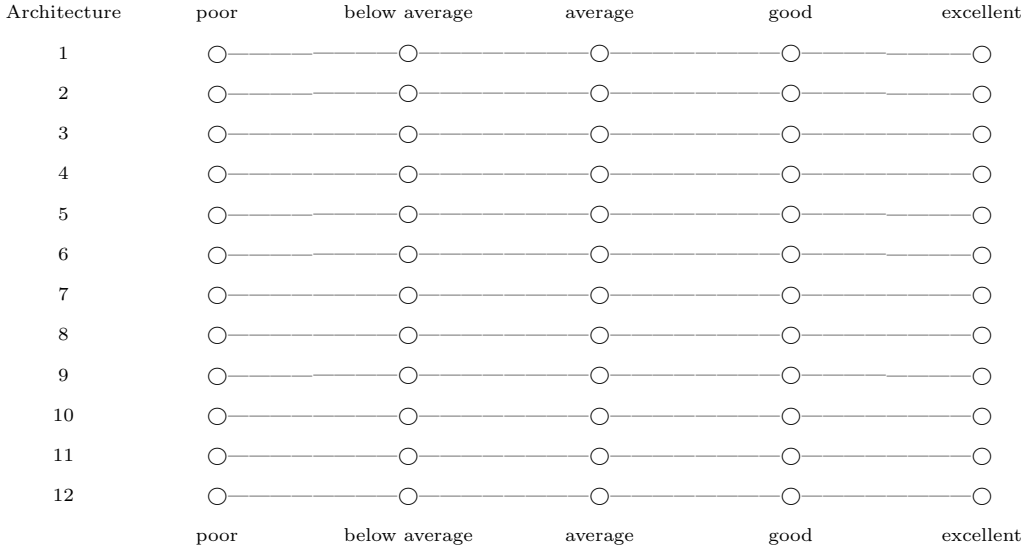

# Diverse Service Level Agreement

Degree to which the system can support a variety of service level agreements to tenants.



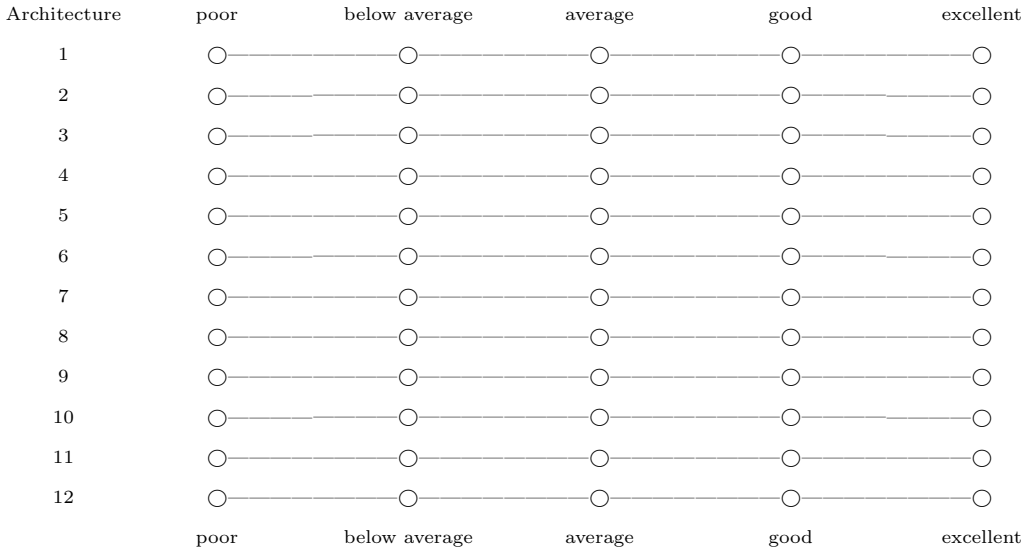
# Software complexity

Degree of the software complexity of the software product if it is developed and implemented in a multi-tenant architecture.




# Monitoring

Degree of ease to which monitoring and controlling tasks can be carried out in the system. Tasks include controlling server availability, user activity, capacity and performance.



# Maintainability

Degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers. Modifications can include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specifications. Modifications include those carried out by specialized support staff, and those carried out by business or operational staff, or end users.
NB Maintainability includes installation of updates and upgrades.

# F Expert Ratings

This section shows the experts ratings of all the multi-tenant architectures against the decision criteria. They are presented in a separate table for each criterion. The first column represents the architectures, they are clickable references to the schema's shown in Section 5.2.1 in the digital version of this work. The five columns after that one represents the 5-point Likert scale items experts used to rate the architectures. The values in these columns define the frequencies of the corresponding ratings. The penultimate column defines the median of the frequencies, denoted by $\mu_{\frac{1}{2}}$. The final column shows the standard deviation of the frequencies, denoted by $\sigma$.

Table 25: Performance Ratings for MTA's on Time Behavior

| | Multi-Tenant Architecture | | Time Behavior Performance | | | | | $\mu_{\frac{1}{2}}$ | $\sigma$ |
|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | | |
| 1 | Dedicated AS | & Dedicated DBS | 0 | 0 | 0 | 1 | 7 | 5.0 | 0.3 |
| 2 | Shared AS | & Dedicated DBS | 0 | 0 | 0 | 6 | 2 | 4.0 | 0.4 |
| 3 | Shared Instance | & Dedicated DBS | 0 | 0 | 3 | 4 | 1 | 4.0 | 0.7 |
| 4 | Dedicated AS | & Shared DBS | 0 | 0 | 2 | 4 | 2 | 4.0 | 0.7 |
| 5 | Shared AS | & Shared DBS | 0 | 0 | 3 | 4 | 1 | 4.0 | 0.7 |
| 6 | Shared Instance | & Shared DBS | 0 | 2 | 4 | 2 | 0 | 3.0 | 0.7 |
| 7 | Dedicated AS | & Shared DB | 0 | 1 | 2 | 3 | 2 | 4.0 | 1.0 |
| 8 | Shared AS | & Shared DB | 0 | 1 | 3 | 3 | 1 | 3.5 | 0.9 |
| 9 | Shared Instance | & Shared DB | 0 | 1 | 6 | 1 | 0 | 3.0 | 0.5 |
| 10 | Dedicated AS | & Shared Schema | 0 | 1 | 3 | 4 | 0 | 3.5 | 0.7 |
| 11 | Shared AS | & Shared Schema | 1 | 2 | 3 | 2 | 0 | 3.0 | 1.0 |
| 12 | Shared Instance | & Shared Schema | 1 | 4 | 2 | 0 | 1 | 2.0 | 1.1 |

Table 26: Performance Ratings for MTA's on Resource Utilization

| | Multi-Tenant Architecture | | Resource Utilization Efficiency | | | | | $\mu_{\frac{1}{2}}$ | $\sigma$ |
|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | | |
| 1 | Dedicated AS | & Dedicated DBS | 3 | 1 | 2 | 0 | 2 | 2.5 | 1.6 |
| 2 | Shared AS | & Dedicated DBS | 1 | 3 | 1 | 3 | 0 | 2.5 | 1.1 |
| 3 | Shared Instance | & Dedicated DBS | 0 | 3 | 3 | 2 | 0 | 3.0 | 0.8 |
| 4 | Dedicated AS | & Shared DBS | 0 | 4 | 2 | 2 | 0 | 2.5 | 0.8 |
| 5 | Shared AS | & Shared DBS | 0 | 0 | 5 | 3 | 0 | 3.0 | 0.5 |
| 6 | Shared Instance | & Shared DBS | 0 | 2 | 3 | 3 | 0 | 3.0 | 0.8 |
| 7 | Dedicated AS | & Shared DB | 0 | 1 | 6 | 1 | 0 | 3.0 | 0.5 |
| 8 | Shared AS | & Shared DB | 0 | 2 | 3 | 2 | 1 | 3.0 | 1.0 |
| 9 | Shared Instance | & Shared DB | 0 | 3 | 0 | 4 | 1 | 4.0 | 1.1 |
| 10 | Dedicated AS | & Shared Schema | 0 | 3 | 4 | 0 | 1 | 3.0 | 0.9 |
| 11 | Shared AS | & Shared Schema | 1 | 1 | 3 | 1 | 2 | 3.0 | 1.3 |
| 12 | Shared Instance | & Shared Schema | 1 | 2 | 0 | 1 | 4 | 4.5 | 1.6 |

Table 27: Performance Ratings for MTA's on Throughput

| | Multi-Tenant Architecture | | Throughput Performance 1 | 2 | 3 | 4 | 5 | $\mu_{\frac{1}{2}}$ | $\sigma$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Dedicated AS | & Dedicated DBS | 0 | 1 | 2 | 1 | 4 | 4.5 | 1.1 |
| 2 | Shared AS | & Dedicated DBS | 0 | 0 | 5 | 3 | 0 | 3.0 | 0.5 |
| 3 | Shared Instance | & Dedicated DBS | 0 | 0 | 6 | 2 | 0 | 3.0 | 0.4 |
| 4 | Dedicated AS | & Shared DBS | 0 | 1 | 2 | 5 | 0 | 4.0 | 0.7 |
| 5 | Shared AS | & Shared DBS | 0 | 0 | 7 | 1 | 0 | 3.0 | 0.3 |
| 6 | Shared Instance | & Shared DBS | 0 | 2 | 4 | 2 | 0 | 3.0 | 0.7 |
| 7 | Dedicated AS | & Shared DB | 0 | 1 | 4 | 3 | 0 | 3.0 | 0.7 |
| 8 | Shared AS | & Shared DB | 0 | 2 | 4 | 1 | 1 | 3.0 | 0.9 |
| 9 | Shared Instance | & Shared DB | 0 | 2 | 4 | 1 | 1 | 3.0 | 0.9 |
| 10 | Dedicated AS | & Shared Schema | 0 | 2 | 3 | 3 | 0 | 3.0 | 0.8 |
| 11 | Shared AS | & Shared Schema | 0 | 2 | 4 | 2 | 0 | 3.0 | 0.7 |
| 12 | Shared Instance | & Shared Schema | 1 | 1 | 4 | 1 | 1 | 3.0 | 1.1 |

Table 28: Performance Ratings for MTA's on Number of Tenants

| | Multi-Tenant Architecture | | Number of Tenants Performance 1 | 2 | 3 | 4 | 5 | $\mu_{\frac{1}{2}}$ | $\sigma$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Dedicated AS | & Dedicated DBS | 6 | 1 | 0 | 1 | 0 | 1.0 | 1.0 |
| 2 | Shared AS | & Dedicated DBS | 0 | 3 | 5 | 0 | 0 | 3.0 | 0.5 |
| 3 | Shared Instance | & Dedicated DBS | 0 | 3 | 4 | 1 | 0 | 3.0 | 0.7 |
| 4 | Dedicated AS | & Shared DBS | 0 | 3 | 4 | 1 | 0 | 3.0 | 0.7 |
| 5 | Shared AS | & Shared DBS | 0 | 4 | 2 | 2 | 0 | 3.5 | 0.8 |
| 6 | Shared Instance | & Shared DBS | 0 | 0 | 3 | 3 | 2 | 4.0 | 0.8 |
| 7 | Dedicated AS | & Shared DB | 0 | 2 | 4 | 2 | 0 | 3.0 | 0.7 |
| 8 | Shared AS | & Shared DB | 0 | 0 | 2 | 3 | 3 | 4.0 | 0.8 |
| 9 | Shared Instance | & Shared DB | 0 | 0 | 1 | 5 | 2 | 4.0 | 0.6 |
| 10 | Dedicated AS | & Shared Schema | 0 | 2 | 4 | 2 | 0 | 3.0 | 0.7 |
| 11 | Shared AS | & Shared Schema | 0 | 0 | 1 | 4 | 3 | 4.0 | 0.7 |
| 12 | Shared Instance | & Shared Schema | 0 | 0 | 2 | 0 | 6 | 5.0 | 0.9 |

Table 29: Performance Ratings for MTA's on Number of End-Users

| | Multi-Tenant Architecture | | Number of end-users Performance 1 | 2 | 3 | 4 | 5 | $\mu_{\frac{1}{2}}$ | $\sigma$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Dedicated AS | & Dedicated DBS | 2 | 2 | 1 | 1 | 2 | 2.5 | 1.5 |
| 2 | Shared AS | & Dedicated DBS | 0 | 3 | 1 | 4 | 0 | 3.5 | 0.9 |
| 3 | Shared Instance | & Dedicated DBS | 0 | 2 | 3 | 3 | 0 | 3.0 | 0.8 |
| 4 | Dedicated AS | & Shared DBS | 0 | 2 | 3 | 3 | 0 | 3.0 | 0.8 |
| 5 | Shared AS | & Shared DBS | 0 | 0 | 4 | 3 | 1 | 3.5 | 0.7 |
| 6 | Shared Instance | & Shared DBS | 0 | 1 | 3 | 3 | 1 | 3.5 | 0.9 |
| 7 | Dedicated AS | & Shared DB | 0 | 1 | 4 | 3 | 0 | 3.0 | 0.7 |
| 8 | Shared AS | & Shared DB | 0 | 1 | 3 | 3 | 1 | 3.5 | 0.9 |
| 9 | Shared Instance | & Shared DB | 0 | 1 | 1 | 5 | 1 | 4.0 | 0.8 |
| 10 | Dedicated AS | & Shared Schema | 0 | 1 | 3 | 4 | 0 | 3.5 | 0.7 |
| 11 | Shared AS | & Shared Schema | 0 | 1 | 2 | 3 | 2 | 4.0 | 1.0 |
| 12 | Shared Instance | & Shared Schema | 0 | 0 | 1 | 4 | 3 | 4.0 | 0.7 |

Table 30: Performance Ratings for MTA's on Availability

| | Multi-Tenant Architecture | | Availability Performance 1 | 2 | 3 | 4 | 5 | $\mu_{\frac{1}{2}}$ | $\sigma$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Dedicated AS | & Dedicated DBS | 0 | 2 | 2 | 0 | 4 | 4.0 | 1.3 |
| 2 | Shared AS | & Dedicated DBS | 0 | 0 | 5 | 3 | 0 | 3.0 | 0.5 |
| 3 | Shared Instance | & Dedicated DBS | 0 | 1 | 5 | 2 | 0 | 3.0 | 0.6 |
| 4 | Dedicated AS | & Shared DBS | 0 | 0 | 5 | 3 | 0 | 3.0 | 0.5 |
| 5 | Shared AS | & Shared DBS | 0 | 0 | 5 | 3 | 0 | 3.0 | 0.5 |
| 6 | Shared Instance | & Shared DBS | 0 | 1 | 4 | 3 | 0 | 3.0 | 0.7 |
| 7 | Dedicated AS | & Shared DB | 0 | 0 | 7 | 1 | 0 | 3.0 | 0.3 |
| 8 | Shared AS | & Shared DB | 0 | 2 | 3 | 3 | 0 | 3.0 | 0.8 |
| 9 | Shared Instance | & Shared DB | 1 | 1 | 3 | 2 | 1 | 3.0 | 1.2 |
| 10 | Dedicated AS | & Shared Schema | 0 | 0 | 6 | 2 | 0 | 3.0 | 0.4 |
| 11 | Shared AS | & Shared Schema | 0 | 2 | 3 | 3 | 0 | 3.0 | 0.8 |
| 12 | Shared Instance | & Shared Schema | 1 | 1 | 3 | 2 | 1 | 3.0 | 1.2 |

Table 31: Performance Ratings for MTA's on Recoverability

| | Multi-Tenant Architecture | | Recoverability Performance 1 | 2 | 3 | 4 | 5 | $\mu_{\frac{1}{2}}$ | $\sigma$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Dedicated AS | & Dedicated DBS | 0 | 1 | 1 | 1 | 5 | 5.0 | 1.1 |
| 2 | Shared AS | & Dedicated DBS | 0 | 1 | 1 | 2 | 4 | 4.5 | 1.1 |
| 3 | Shared Instance | & Dedicated DBS | 0 | 1 | 1 | 2 | 4 | 4.5 | 1.1 |
| 4 | Dedicated AS | & Shared DBS | 0 | 1 | 2 | 4 | 1 | 4.0 | 0.9 |
| 5 | Shared AS | & Shared DBS | 0 | 1 | 2 | 4 | 1 | 4.0 | 0.9 |
| 6 | Shared Instance | & Shared DBS | 0 | 1 | 2 | 4 | 1 | 4.0 | 0.9 |
| 7 | Dedicated AS | & Shared DB | 0 | 2 | 5 | 1 | 0 | 3.0 | 0.6 |
| 8 | Shared AS | & Shared DB | 0 | 2 | 5 | 1 | 0 | 3.0 | 0.6 |
| 9 | Shared Instance | & Shared DB | 0 | 2 | 5 | 1 | 0 | 3.0 | 0.6 |
| 10 | Dedicated AS | & Shared Schema | 0 | 6 | 2 | 0 | 0 | 2.0 | 0.4 |
| 11 | Shared AS | & Shared Schema | 1 | 5 | 2 | 0 | 0 | 2.0 | 0.6 |
| 12 | Shared Instance | & Shared Schema | 1 | 5 | 1 | 0 | 1 | 2.0 | 1.1 |

Table 32: Performance Ratings for MTA's on Confidentiality

| | Multi-Tenant Architecture | | Confidentiality Performance 1 | 2 | 3 | 4 | 5 | $\mu_{\frac{1}{2}}$ | $\sigma$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Dedicated AS | & Dedicated DBS | 0 | 0 | 0 | 2 | 6 | 5.0 | 0.4 |
| 2 | Shared AS | & Dedicated DBS | 0 | 0 | 0 | 4 | 4 | 4.5 | 0.5 |
| 3 | Shared Instance | & Dedicated DBS | 0 | 0 | 3 | 3 | 2 | 4.0 | 0.8 |
| 4 | Dedicated AS | & Shared DBS | 0 | 0 | 1 | 5 | 2 | 4.0 | 0.6 |
| 5 | Shared AS | & Shared DBS | 0 | 0 | 1 | 5 | 2 | 4.0 | 0.6 |
| 6 | Shared Instance | & Shared DBS | 0 | 0 | 3 | 4 | 1 | 4.0 | 0.7 |
| 7 | Dedicated AS | & Shared DB | 0 | 1 | 3 | 3 | 1 | 3.5 | 0.9 |
| 8 | Shared AS | & Shared DB | 0 | 1 | 4 | 2 | 1 | 3.0 | 0.9 |
| 9 | Shared Instance | & Shared DB | 0 | 1 | 6 | 0 | 1 | 3.0 | 0.8 |
| 10 | Dedicated AS | & Shared Schema | 1 | 4 | 3 | 0 | 0 | 2.0 | 0.7 |
| 11 | Shared AS | & Shared Schema | 1 | 5 | 2 | 0 | 0 | 2.0 | 0.6 |
| 12 | Shared Instance | & Shared Schema | 2 | 6 | 0 | 0 | 0 | 2.0 | 0.4 |

Table 33: Performance Ratings for MTA's on Integrity

| | Multi-Tenant Architecture | | Integrity Performance 1 | 2 | 3 | 4 | 5 | $\mu_{\frac{1}{2}}$ | $\sigma$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Dedicated AS | & Dedicated DBS | 1 | 0 | 0 | 3 | 4 | 4.5 | 1.3 |
| 2 | Shared AS | & Dedicated DBS | 0 | 1 | 1 | 6 | 0 | 4.0 | 0.7 |
| 3 | Shared Instance | & Dedicated DBS | 0 | 2 | 4 | 2 | 0 | 3.0 | 0.7 |
| 4 | Dedicated AS | & Shared DBS | 0 | 1 | 1 | 5 | 1 | 4.0 | 0.8 |
| 5 | Shared AS | & Shared DBS | 0 | 0 | 4 | 4 | 0 | 3.5 | 0.5 |
| 6 | Shared Instance | & Shared DBS | 0 | 1 | 7 | 0 | 0 | 3.0 | 0.3 |
| 7 | Dedicated AS | & Shared DB | 0 | 1 | 3 | 3 | 1 | 3.5 | 0.9 |
| 8 | Shared AS | & Shared DB | 0 | 1 | 5 | 2 | 0 | 3.0 | 0.6 |
| 9 | Shared Instance | & Shared DB | 0 | 2 | 6 | 0 | 0 | 3.0 | 0.4 |
| 10 | Dedicated AS | & Shared Schema | 0 | 1 | 5 | 1 | 1 | 3.0 | 0.8 |
| 11 | Shared AS | & Shared Schema | 0 | 4 | 2 | 2 | 0 | 2.5 | 0.8 |
| 12 | Shared Instance | & Shared Schema | 2 | 2 | 4 | 0 | 0 | 2.5 | 0.8 |

Table 34: Performance Ratings for MTA's on Authenticity

| | Multi-Tenant Architecture | | Authenticity Performance 1 | 2 | 3 | 4 | 5 | $\mu_{\frac{1}{2}}$ | $\sigma$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Dedicated AS | & Dedicated DBS | 1 | 0 | 2 | 1 | 4 | 4.5 | 1.4 |
| 2 | Shared AS | & Dedicated DBS | 0 | 0 | 4 | 3 | 1 | 3.5 | 0.7 |
| 3 | Shared Instance | & Dedicated DBS | 0 | 2 | 4 | 2 | 0 | 3.0 | 0.7 |
| 4 | Dedicated AS | & Shared DBS | 1 | 0 | 3 | 3 | 1 | 3.5 | 1.1 |
| 5 | Shared AS | & Shared DBS | 0 | 0 | 5 | 2 | 1 | 3.0 | 0.7 |
| 6 | Shared Instance | & Shared DBS | 0 | 1 | 4 | 2 | 1 | 3.0 | 0.9 |
| 7 | Dedicated AS | & Shared DB | 0 | 1 | 2 | 4 | 1 | 4.0 | 0.9 |
| 8 | Shared AS | & Shared DB | 0 | 0 | 5 | 3 | 0 | 3.0 | 0.5 |
| 9 | Shared Instance | & Shared DB | 0 | 2 | 4 | 2 | 0 | 3.0 | 0.7 |
| 10 | Dedicated AS | & Shared Schema | 0 | 1 | 5 | 2 | 0 | 3.0 | 0.6 |
| 11 | Shared AS | & Shared Schema | 0 | 2 | 4 | 2 | 0 | 3.0 | 0.7 |
| 12 | Shared Instance | & Shared Schema | 3 | 0 | 4 | 0 | 1 | 3.0 | 1.3 |

Table 35: Performance Ratings for MTA's on Migration

| | Multi-Tenant Architecture | | Migration Performance 1 | 2 | 3 | 4 | 5 | $\mu_{\frac{1}{2}}$ | $\sigma$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Dedicated AS | & Dedicated DBS | 1 | 0 | 0 | 1 | 6 | 5.0 | 1.3 |
| 2 | Shared AS | & Dedicated DBS | 0 | 1 | 0 | 2 | 5 | 5.0 | 1.0 |
| 3 | Shared Instance | & Dedicated DBS | 0 | 0 | 1 | 2 | 5 | 5.0 | 0.7 |
| 4 | Dedicated AS | & Shared DBS | 0 | 1 | 0 | 3 | 4 | 4.5 | 1.0 |
| 5 | Shared AS | & Shared DBS | 0 | 0 | 2 | 2 | 4 | 4.5 | 0.8 |
| 6 | Shared Instance | & Shared DBS | 0 | 0 | 1 | 3 | 4 | 4.5 | 0.7 |
| 7 | Dedicated AS | & Shared DB | 0 | 1 | 2 | 5 | 0 | 4.0 | 0.7 |
| 8 | Shared AS | & Shared DB | 0 | 0 | 3 | 5 | 0 | 4.0 | 0.5 |
| 9 | Shared Instance | & Shared DB | 0 | 0 | 3 | 5 | 0 | 4.0 | 0.5 |
| 10 | Dedicated AS | & Shared Schema | 1 | 2 | 4 | 1 | 0 | 3.0 | 0.9 |
| 11 | Shared AS | & Shared Schema | 1 | 1 | 5 | 1 | 0 | 3.0 | 0.8 |
| 12 | Shared Instance | & Shared Schema | 1 | 1 | 5 | 0 | 1 | 3.0 | 1.1 |

Table 36: Performance Ratings for MTA's on Deployment Time

| | Multi-Tenant Architecture | | Deployment Time Performance 1 | 2 | 3 | 4 | 5 | $\mu_{\frac{1}{2}}$ | $\sigma$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Dedicated AS | & Dedicated DBS | 4 | 1 | 2 | 0 | 1 | 1.5 | 1.4 |
| 2 | Shared AS | & Dedicated DBS | 0 | 3 | 3 | 2 | 0 | 3.0 | 0.8 |
| 3 | Shared Instance | & Dedicated DBS | 0 | 1 | 4 | 2 | 1 | 3.0 | 0.9 |
| 4 | Dedicated AS | & Shared DBS | 1 | 3 | 3 | 1 | 0 | 2.5 | 0.9 |
| 5 | Shared AS | & Shared DBS | 0 | 2 | 2 | 4 | 0 | 3.5 | 0.8 |
| 6 | Shared Instance | & Shared DBS | 0 | 0 | 1 | 6 | 1 | 4.0 | 0.5 |
| 7 | Dedicated AS | & Shared DB | 1 | 2 | 5 | 0 | 0 | 3.0 | 0.7 |
| 8 | Shared AS | & Shared DB | 0 | 2 | 1 | 4 | 1 | 4.0 | 1.0 |
| 9 | Shared Instance | & Shared DB | 0 | 0 | 2 | 3 | 3 | 4.0 | 0.8 |
| 10 | Dedicated AS | & Shared Schema | 1 | 1 | 6 | 0 | 0 | 3.0 | 0.7 |
| 11 | Shared AS | & Shared Schema | 0 | 1 | 2 | 4 | 1 | 4.0 | 0.9 |
| 12 | Shared Instance | & Shared Schema | 0 | 1 | 0 | 1 | 6 | 5.0 | 1.0 |

Table 37: Performance Ratings for MTA's on Variability

| | Multi-Tenant Architecture | | Variability Performance 1 | 2 | 3 | 4 | 5 | $\mu_{\frac{1}{2}}$ | $\sigma$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Dedicated AS | & Dedicated DBS | 0 | 0 | 0 | 1 | 7 | 5.0 | 0.3 |
| 2 | Shared AS | & Dedicated DBS | 0 | 0 | 1 | 5 | 2 | 4.0 | 0.6 |
| 3 | Shared Instance | & Dedicated DBS | 0 | 5 | 1 | 2 | 0 | 2.0 | 0.9 |
| 4 | Dedicated AS | & Shared DBS | 0 | 0 | 0 | 3 | 5 | 5.0 | 0.5 |
| 5 | Shared AS | & Shared DBS | 0 | 0 | 2 | 4 | 2 | 4.0 | 0.7 |
| 6 | Shared Instance | & Shared DBS | 0 | 5 | 2 | 1 | 0 | 2.0 | 0.7 |
| 7 | Dedicated AS | & Shared DB | 0 | 0 | 2 | 2 | 4 | 4.5 | 0.8 |
| 8 | Shared AS | & Shared DB | 0 | 0 | 4 | 2 | 2 | 3.5 | 0.8 |
| 9 | Shared Instance | & Shared DB | 0 | 6 | 1 | 1 | 0 | 2.0 | 0.7 |
| 10 | Dedicated AS | & Shared Schema | 0 | 4 | 2 | 1 | 1 | 2.5 | 1.1 |
| 11 | Shared AS | & Shared Schema | 0 | 6 | 0 | 2 | 0 | 2.0 | 0.9 |
| 12 | Shared Instance | & Shared Schema | 5 | 2 | 1 | 0 | 0 | 1.0 | 0.7 |

Table 38: Performance Ratings for MTA's on Diverse SLA

| | Multi-Tenant Architecture | | Diverse SLA Performance 1 | 2 | 3 | 4 | 5 | $\mu_{\frac{1}{2}}$ | $\sigma$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Dedicated AS | & Dedicated DBS | 0 | 0 | 0 | 1 | 7 | 5.0 | 0.3 |
| 2 | Shared AS | & Dedicated DBS | 0 | 0 | 1 | 5 | 2 | 4.0 | 0.6 |
| 3 | Shared Instance | & Dedicated DBS | 0 | 3 | 3 | 1 | 1 | 3.0 | 1.0 |
| 4 | Dedicated AS | & Shared DBS | 0 | 0 | 0 | 7 | 1 | 4.0 | 0.3 |
| 5 | Shared AS | & Shared DBS | 0 | 0 | 4 | 3 | 1 | 3.5 | 0.7 |
| 6 | Shared Instance | & Shared DBS | 0 | 4 | 3 | 1 | 0 | 2.5 | 0.7 |
| 7 | Dedicated AS | & Shared DB | 0 | 0 | 3 | 5 | 0 | 4.0 | 0.5 |
| 8 | Shared AS | & Shared DB | 0 | 0 | 6 | 2 | 0 | 3.0 | 0.4 |
| 9 | Shared Instance | & Shared DB | 0 | 3 | 5 | 0 | 0 | 3.0 | 0.5 |
| 10 | Dedicated AS | & Shared Schema | 0 | 1 | 4 | 3 | 0 | 3.0 | 0.7 |
| 11 | Shared AS | & Shared Schema | 0 | 4 | 4 | 0 | 0 | 2.5 | 0.5 |
| 12 | Shared Instance | & Shared Schema | 3 | 5 | 0 | 0 | 0 | 2.0 | 0.5 |

Table 39: Performance Ratings for MTA's on Software Complexity

| | Multi-Tenant Architecture | | Software Complexity Performance 1 | 2 | 3 | 4 | 5 | $\mu_{\frac{1}{2}}$ | $\sigma$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Dedicated AS | & Dedicated DBS | 0 | 0 | 0 | 2 | 6 | 5.0 | 0.4 |
| 2 | Shared AS | & Dedicated DBS | 0 | 0 | 1 | 3 | 4 | 4.5 | 0.7 |
| 3 | Shared Instance | & Dedicated DBS | 0 | 1 | 1 | 6 | 0 | 4.0 | 0.7 |
| 4 | Dedicated AS | & Shared DBS | 0 | 0 | 1 | 3 | 4 | 4.5 | 0.7 |
| 5 | Shared AS | & Shared DBS | 0 | 0 | 2 | 2 | 4 | 4.5 | 0.8 |
| 6 | Shared Instance | & Shared DBS | 0 | 1 | 3 | 4 | 0 | 3.5 | 0.7 |
| 7 | Dedicated AS | & Shared DB | 0 | 1 | 1 | 4 | 2 | 4.0 | 0.9 |
| 8 | Shared AS | & Shared DB | 0 | 0 | 3 | 3 | 2 | 4.0 | 0.8 |
| 9 | Shared Instance | & Shared DB | 0 | 1 | 5 | 2 | 0 | 3.0 | 0.6 |
| 10 | Dedicated AS | & Shared Schema | 0 | 4 | 3 | 1 | 0 | 2.5 | 0.7 |
| 11 | Shared AS | & Shared Schema | 0 | 4 | 3 | 1 | 0 | 2.5 | 0.7 |
| 12 | Shared Instance | & Shared Schema | 2 | 3 | 2 | 1 | 0 | 2.0 | 1.0 |

Table 40: Performance Ratings for MTA's on Monitoring

| | Multi-Tenant Architecture | | Monitoring Performance 1 | 2 | 3 | 4 | 5 | $\mu_{\frac{1}{2}}$ | $\sigma$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Dedicated AS | & Dedicated DBS | 5 | 2 | 0 | 0 | 1 | 1.0 | 1.3 |
| 2 | Shared AS | & Dedicated DBS | 0 | 4 | 3 | 1 | 0 | 2.5 | 0.7 |
| 3 | Shared Instance | & Dedicated DBS | 0 | 1 | 7 | 0 | 0 | 3.0 | 0.3 |
| 4 | Dedicated AS | & Shared DBS | 1 | 3 | 4 | 0 | 0 | 2.5 | 0.7 |
| 5 | Shared AS | & Shared DBS | 0 | 1 | 4 | 3 | 0 | 3.0 | 0.7 |
| 6 | Shared Instance | & Shared DBS | 0 | 0 | 6 | 2 | 0 | 3.0 | 0.4 |
| 7 | Dedicated AS | & Shared DB | 0 | 3 | 4 | 1 | 0 | 3.0 | 0.7 |
| 8 | Shared AS | & Shared DB | 0 | 0 | 3 | 4 | 1 | 4.0 | 0.7 |
| 9 | Shared Instance | & Shared DB | 0 | 0 | 1 | 5 | 2 | 4.0 | 0.6 |
| 10 | Dedicated AS | & Shared Schema | 0 | 3 | 4 | 0 | 1 | 3.0 | 0.9 |
| 11 | Shared AS | & Shared Schema | 0 | 1 | 0 | 5 | 2 | 4.0 | 0.9 |
| 12 | Shared Instance | & Shared Schema | 0 | 1 | 0 | 2 | 5 | 5.0 | 1.0 |

Table 41: Performance Ratings for MTA's on Maintainability

| | Multi-Tenant Architecture | | Maintainability Performance 1 | 2 | 3 | 4 | 5 | $\mu_{\frac{1}{2}}$ | $\sigma$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Dedicated AS | & Dedicated DBS | 4 | 3 | 1 | 0 | 1 | 1.5 | 1.3 |
| 2 | Shared AS | & Dedicated DBS | 1 | 3 | 3 | 1 | 0 | 2.5 | 0.9 |
| 3 | Shared Instance | & Dedicated DBS | 0 | 2 | 4 | 2 | 0 | 3.0 | 0.7 |
| 4 | Dedicated AS | & Shared DBS | 2 | 3 | 2 | 0 | 1 | 2.0 | 1.2 |
| 5 | Shared AS | & Shared DBS | 0 | 2 | 4 | 2 | 0 | 3.0 | 0.7 |
| 6 | Shared Instance | & Shared DBS | 0 | 0 | 4 | 4 | 0 | 3.5 | 0.5 |
| 7 | Dedicated AS | & Shared DB | 0 | 4 | 3 | 1 | 0 | 2.5 | 0.7 |
| 8 | Shared AS | & Shared DB | 0 | 1 | 2 | 5 | 0 | 4.0 | 0.7 |
| 9 | Shared Instance | & Shared DB | 0 | 0 | 2 | 4 | 2 | 4.0 | 0.7 |
| 10 | Dedicated AS | & Shared Schema | 0 | 1 | 5 | 2 | 0 | 3.0 | 0.6 |
| 11 | Shared AS | & Shared Schema | 0 | 0 | 1 | 5 | 2 | 4.0 | 0.6 |
| 12 | Shared Instance | & Shared Schema | 0 | 0 | 0 | 2 | 6 | 5.0 | 0.4 |