Multi-Dimensional Server Consolidation for Commercial Off-the-Shelf Enterprise Applications Using Shared Performance Counters

Dissertation

zur Erlangung des akademischen Grades

Doktoringenieur (Dr.-Ing.)

angenommen durch die Fakultät für Informatik der
Otto-von-Guericke-Universität Magdeburg

von **Hendrik Müller, M. Sc.**
geboren am 20.06.1987 in Schwerin

Gutachter:
Prof. Dr. Klaus Turowski
Prof. Dr. Gunter Saake
Prof. Dr. Steffen Becker

Magdeburg, den 16. September 2019

Abstract

Internal and external IT service providers increasingly use commercial-off-the-shelf software to support business processes. As these applications are continuously monitored, emerging log data follows a standardized format and, in its internal logic, is comparable across organizational boundaries. Resulting automation potential leads to cost savings and quality improvements in the context of capacity management. These objectives are examined by the thesis at hand using the capacity management method PPSS (Performance prediction supported service placement), which is designed to address the automation potential for server consolidation scenarios.

For this purpose, a placement problem is formulated that meets the special requirements of enterprise applications. The objective of the problem is to save operations costs by minimizing the required capacity. Solution quality is further affected by the compliance with a variety of constraints. Four heuristics, two metaheuristics, and two hybrid algorithms are evaluated in 12,384 field experiments with respect to their solution quality. As the number of constraints increases, genetic algorithms tend to identify solutions of the highest relative quality. In all scenarios, more than 20% of the original server capacity can be saved on average while complying with the given constraints.

Too aggressive capacity reduction, however, increases the risk of violating performance-related service level agreements and entails the payment of penalty costs. The prediction of transactional response times that are expected from a solution candidate enables to estimate the amount of such penalty costs. At the same time, solution credibility is increased. For this purpose, PPSS integrates the use of black-box approaches, which are based on machine learning and keep personnel costs low. In addition, the tested techniques benefit from the cross-organizational integration of log data due to large volume and variety of the observations on which the learning process is based. Using the example of a widespread standard transaction, both Random forests and Boosted trees prove to be suitable methods for predicting the mean response times of dialog steps. Boosted trees show mean absolute percent errors between 19% and 30% across additional test cases on frequently used business transactions.

A case study demonstrates the utility of the method. Here, alternative solution candidates of the placement problem are analyzed with regard to their total costs. These consist of operations costs and penalty costs. Using the predicted response times and an exemplary service level agreement, the amount of expected penalty costs can be estimated for alternative load scenarios. If future load probabilities are known, a single solution can be recommended to minimize the total costs.

PPSS is technically enabled by a performance knowledge base consisting of three layers which cover presentation, analysis, and data. The knowledge base is implemented to evaluate the research artifact in a real environment. Selected process steps are supported on the presentation level by a graphical user interface. This interface is offered to a group of test users from two different data centers as part of a pilot operations phase. The user feedback proves the utility and indicates cost savings of the method when compared to existing approaches.

Abstract in German

Interne und externe IT-Dienstleister setzen vermehrt betriebliche Standardsoftware ein, um Geschäftsprozesse abzudecken. Im Rahmen der kontinuierlichen Überwachung dieser Anwendungssysteme werden daher Logdaten generiert, die einem standardisierten Format entsprechen und in ihrer internen Logik über Organisationsgrenzen hinweg vergleichbar sind. Hieraus leitet sich Automatisierungspotenzial ab, welches im Rahmen des Kapazitätsmanagements zu Kostenersparnissen und Qualitätsverbesserungen führt. Die vorliegende Arbeit untersucht diese Zielstellung anhand der Kapazitätsmanagement-Methode PPSS (Performance prediction supported service placement), die dieses Potenzial für den Anwendungsfall der Serverkonsolidierung addressiert.

Zu diesem Zwecke wird ein Platzierungsproblem formuliert, das den besonderen Anforderungen betrieblicher Anwendungssysteme gerecht wird. Ziel des Problems ist es, Betriebskosten durch Minimierung der Kapazität einzusparen, wobei die Konformität mit einer Vielzahl von Nebenbedingungen die Lösungsqualität beeinflusst. Vier Heuristiken, zwei Metaheuristiken und zwei hybride Algorithmen werden in insgesamt 12.384 Feldexperimenten hinsichtlich Ihrer Lösungsqualität evaluiert. Mit steigendem Ausmaß der Nebenbedingungen identifizieren genetische Algorithmen Lösungen von höchster, relativer Qualität. In allen Szenarien kann bei gleichzeitiger Konformität mit den gegebenen Nebenbedingungen im Mittel mehr als 20% der ursprünglichen Serverkapazität eingespart werden.

Eine zu starke Reduzierung der Kapazität erhöht jedoch das Risiko, Dienstgütevereinbarungen zu verletzen, was zu Strafzahlungen führt. Eine Vorhersage der zu erwartenden, transaktionalen Antwortzeiten eines Lösungskandidaten ermöglicht es, das Ausmaß der Strafzahlungen zu schätzen und damit das Vertrauen in die Lösung zu erhöhen. PPSS integriert zu diesem Zweck den Einsatz so genannter Black-Box-Ansätze, welche auf maschinellem Lernen basieren und damit die Personalkosten gering halten. Die getesteten Verfahren profitieren zudem von der organisationsübergreifenden Integration der Logdaten. Dies erhöht den Umfang und die Vielfalt der Beobachtungen, die dem Lernprozess zugrunde gelegt werden. Am Beispiel einer weit verbreiteten Standardtransaktion erweisen sich sowohl Random Forests als auch Boosted Trees als geeignete Verfahren für die Vorhersage mittlerer Antwortzeiten von Dialogschritten. Letzteres führte auch bei weiteren Tests mit häufig genutzten Geschäftstransaktionen zu Vorhersagen mit einer mittleren prozentualen Abweichung zwischen 19% und 30%. Eine Fallstudie demonstriert die Nutzbarkeit der Methode. Hierbei werden alternative Lö-sungskandidaten des Platzierungsproblems hinsichtlich ihrer Gesamtkosten analysiert. Diese setzen sich aus Betriebskosten und Strafzahlungen zusammen. Anhand der Modell-Vorhersagen und einer beispielhaften Dienstgütevereinbarung kann die Höhe der erwarteten Strafzahlungen für alternative Lastszenarien geschätzt werden. Dies ermöglicht, bei Kenntnis zukünftiger Lastwahrscheinlichkeiten, eine Lösungsempfehlung, die die Gesamtkosten minimiert.

PPSS wird technisch durch eine Performance-Wissensdatenbank ermöglicht, die aus den Ebenen Präsentation, Analyse, und Datenhaltung besteht. Für die Evaluierung in einer realen Umgebung wird diese Wissensdatenbank implementiert. Ausgewählte Prozessschritte werden dabei auf der Präsentationsebene durch eine grafische Oberfläche unterstützt. Diese wird im Rahmen eines Pilotbetriebes einer Gruppe von Testbenutzern aus zwei unterschiedlichen Rechenzentren zur Verfügung gestellt. Das Benutzer-Feedback belegt die Nützlichkeit und weist auf Kostenersparnisse hin, die im Vergleich zu bisherigen Ansätzen entstehen.

Contents

List of Figures

List of Tables

List of Abbreviations

ACO ..... Ant colony optimization

APM ..... Application performance monitoring

APM-KB ..... Application performance monitoring knowledge base

ASP ..... Application service provider

ASUM-DM ..... Analytics solutions unified method for data mining

AWS ..... Amazon web services

BFD ..... Best-fit decreasing

CMIS ..... Capacity management information system

COTS ..... Commercial off-the-shelf

CPU ..... Central processing unit

CRISP-DM ..... Cross-industry standard process for data mining

EA ..... Enterprise application

EPA ..... Environmental Protection Agency

ERP ..... Enterprise resource planning

FFD ..... First-fit decreasing

GA ..... Genetic algorithm

GGA ..... Grouping genetic algorithm

GUI ..... Graphical user interface

HA ..... High availability

HPO ..... Hyperparameter optimization

I/O ..... Input/output

IaaS ..... Infrastructure as a service

ICT ..... Information and communications technology

IS ..... Information system

ISO ..... International organization for standardization

IT ..... Information technology

ITIL ..... IT Infrastructure Library

ITSM ..... IT Service Management

JSON ..... JavaScript object notation

LOS ..... Levels of service

LP ..... Linear programming

MAM ..... Multi-objective solved as mono-objective

MAPE-K ..... Monitor Analyze Plan Execute Knowledge

MIT ..... Massachusetts Institute of Technology

ML ..... Machine learning

MTTF ..... Mean time to failure

MTTR ..... Mean time to recovery

NAS ..... Network attached storage

OS ..... Operating system

PF ..... Penalty factor

PK ..... Polynomial kernel

PM ..... Physical machine

PMO ..... Pure multi-objective

PPSS ..... Performance prediction supported service placement

QDH ..... Quality-driven heuristic

QoE ..... Quality of experience

QoS ..... Quality of service

RBF ..... Radial basis function

RF ..... Random forest

RQ ..... Research question

SAN ..... Storage area network

SAPS ..... SAP application performance standard

SD ..... Sales and distribution

SDK ..... Software development toolkit

SDLC ..... software development life-cycle

SLA ..... Service level agreement

SLM ..... Service level management

SPE ..... Software performance engineering

SPP ..... Servie placement problem

SQL ..... Structured query language

SSAP ..... Static server allocation problem

SSAPv ..... Static server allocation problem with variable workload

SVM ..... support vector machines

TCO ..... Total cost of ownership

UML ..... Unified modeling language

VBP ..... Vector bin packing

VM ..... Virtual machine

VMP ..... Virtual machine placement

1 Introduction

"We're entering a new world in which data may be more important than software."
Tim O'Reilly, CEO of O'Reilly Media

1.1 Motivation

The incredible growth of the world's dependence on computers and on-line services (Johnson and Marker, 2009) will likely increase data center's electricity demand in the future. Both public organizations and private corporations face rapidly growing information processing requirements in order to support digital services in various industrial sectors such as manufacturing, finance, transportation, or housing (Goudarzi et al., 2012). Recent trends in the worldwide labor market reflect this development under the term digital transformation (Matt et al., 2015; Gimpel and Röglinger, 2015), which current hypes such as big data, machine learning, and the internet of things contribute to. In 2017, Frey and Osborne investigate the probability of job automation (which they refer to as computerization) for 702 occupations and estimated impacts on the US labor market. According to their study, 47% of total US employment is at high risk to be automated by 2033. While "routine tasks involving explicit rule-based activities" show highest risk according to related literature and common sense, they additionally identified domains that require "non-routine cognitive tasks" to be entered by algorithms for big data in the future (Frey and Osborne, 2017, p. 44). (Ironically, machine learning techniques were used to classify occupations into groups of computerization risk.) The global consequences across whole industries and related occupations are severe; many experts proclaim the fourth industrial revolution which relies heavily on connected devices and data analytics (Lasi et al., 2014; Lee et al., 2014; Rüßmann et al., 2015). In a local study, the Centre for Social Research Halle (ZSH) in Saxony-Anhalt identified processes of the ICT (Information and communications technology) sector itself to be most affected by the ongoing digitization (Heyme and Menge, 2017), resulting in some sort of digitization of IT (Information technology). While this may sound paradoxical at first, the progress becomes historically reasonable: basic principles such as continuous improvement and modularization were adapted from the manufacturing sector in the course of the *Industrialization of IT* (Walter et al., 2007). As manufacturing faces the next revolution under the term *Industry 4.0*, it may be reasonable to apply the underlying principles of interconnection, automation, and data analysis also to IT operations. Thinking ahead with respect to the ongoing transformation, IT processes which strongly rely on human experts will become subject to higher degrees of

digitization in the future. A prominent example, central to IT operations, is the process of managing IT capacity; it is known to be a complex task which involves high manual effort (Cherkasova and Rolia, 2006) and, in many cases, large amounts of historical measurement data. This creates an excellent scenario to leverage the potential of computational intelligence and data analysis. In turn, the application area of the capacity management process is highly business-critical: IT resources being managed are utilized by enterprise applications (EA) which support corporate business functions and business tasks. The fulfillment of non-functional software requirements such as availability and performance for enterprise applications is crucial to the successful and effective execution of business processes (Grinshpan, 2012; Beloglazov et al., 2012). In fact, the negative consequences of performance failures may include damaged customer relations, lost income, increased maintenance costs, delayed project schedules, and project failures (Tudenhöfner, 2011).

The process of managing capacity, redesigned in the course of the digital transformation, is expected to address optimization potential more effectively. Accompanying saving potential must be exploited as information technology, on the other hand, represents a major cost factor for enterprises. According to the worldwide IT spending forecast by Gartner, the overall IT costs will grow by 3.2% to a total of 3.8 trillion dollars in 2019. Particularly, costs for enterprise software will grow by 8.3% (Gartner, 2018b). The total cost of ownership (TCO) for data centers is often dominated by energy costs, which have dramatically increased in the recent decade (Johnson and Marker, 2009; Filani et al., 2008; Orgerie et al., 2014). On a global scale, 1.3% of the worldwide electricity consumption could be assigned to data centers in 2011 (Koomey, 2011). This estimation is consistent with Speitkamp and Bichler (2010) who state that approximately 0.5% of global CO_2 emissions can be assigned to running servers. In some cases, 40-50% of the total data center operational budget is spent on energy costs for IT components (Filani et al., 2008). Although the energy efficiency of certain hardware components has been improved in recent years, at the same time, the overall energy consumption of data centers has increased by 56% from 2005 to 2010 (Koomey, 2011; Splieth et al., 2015; Müller et al., 2016b). Minas and Ellison (2009) state that energy expenses will soon equal hardware costs if computed over a period of three years. This causes an "invisible crisis" in data centers (Johnson and Marker, 2009; Brill, 2007), which, naturally, raises additional concerns about pollution, carbon emissions, and environmental sustainability. Accordingly, the current top three global risks, in terms of impact and likelihood, are environmental: "Extreme weather events", "natural disasters", and the "failure of climate-change mitigation and adaption" (Collins, 2018, Fig. 1). Hence, governmental institutions such as the U.S. EPA and the European Commission have recognized the need to maximize data center's energy efficiency in order to extenuate resulting greenhouse gas emissions and keep disruptive effects on electricity infrastructure under control (Johnson and Marker, 2009).

In short, enterprise applications and their performance are increasingly vital to business continuity and, at the same time, cause rapidly growing expenses. Therefore, IT Service Management (ITSM) frameworks such as the IT Infrastructure Library (ITIL)

and International Organization for Standardization (ISO) 20000 embed the task of balancing performance and operational costs into the capacity management process. Since EAs need to permanently adjust to individual, ever-changing business environments (Grinshpan, 2012) which they support, capacity management requires to continually optimize resource provisioning in an *as little as possible, as much as necessary*-manner in order to be competitive. Therefore, the provisioning of minimum hardware capacity that still ensures cost-effective operations, aligned to given service levels and business constraints, is at the heart of capacity management decisions. According to several studies, however, average server utilization levels usually vary between 10 and 20 percent in practice (Müller et al., 2016b; Beloglazov and Buyya, 2010a; Speitkamp and Bichler, 2010). In June 2016, the *United States Data Center Energy Usage Report* forecasted the average utilization of active volume servers to be approximately 15% for internal data centers and 25% for data centers of service providers in 2020 (Shehabi et al., 2016). Such low utilization rates severely affect energy usage since servers do not run energy-proportional. In fact, idle resources consume up to 70 percent of their peak power (Beloglazov and Buyya, 2010a; Shehabi et al., 2016; Barroso and Hölzle, 2007). Therefore, raising utilization levels barely effects power consumption of individual servers but minimizes the total number of servers needed. On this matter, recent studies state that up to 30 percent of all servers operated in the US were not used in 6 months or more and, thus, termed as *comatose* (Koomey and Taylor, 2015; Kaplan et al., 2008). Although several approaches exist to address existing consolidation potential, those appear to be limited in terms of effectiveness and applicability. In practice, various dependencies and existing requirements related to, e.g., compatibility, fault tolerance, licensing, or security, limit degrees of freedom in service placement and, therefore, reduce the addressable optimization potential (Shaw, 2004; Hyser et al., 2007; Dang and Hermenier, 2013). Such constraints are widely disregarded by existing consolidation algorithm implementations although known to be mandatory (Speitkamp and Bichler, 2010; López-Pires and Báran, 2015; Pires and Barán, 2013). Furthermore, the credibility of calculated solution candidates in many cases remains questionable and is rather a matter of trust or additional expert consultation. Solution feasibility is usually evaluated from a strong technical perspective with respect to capacity savings. As stated by Stillwell et al., "in practice, however, resource management objectives are expressed based on higher level metrics that are related to user concerns, such as service response time or throughput"(Stillwell et al., 2010, p. 6). Likewise, service level agreements (SLA) are typically expressed from a user perspective to ensure measurability at service consumer site. Most server consolidation approaches consider those higher level metrics only indirectly by mapping them to resource capacity values in a reasonable way (Stillwell et al., 2010). Yet this high level of abstraction does not allow for quantifying response times or other related user-level metrics and strongly limits solution evaluation with respect to given SLAs.

Hence, to provide confident decision support as part of the capacity management process, the performance of an EA needs to be predicted and evaluated considering dif-

ferent design alternatives. As stated by Brunnert and Krcmar (2015), ideally, a system which matches the final production environment would be desirable during capacity testing. Yet this precondition is rarely fulfilled and smaller scale systems are hardly comparable (Brunnert and Krcmar, 2015). "We cannot do anything about performance until we have something running to measure" is a statement from practice that was identified by Tudenhöfner (2011, p. 11) as a frequent argument for managing performance reactively. However, one can argue that, due to the intensive use of commercial off-the-shelf (COTS) software in the field of enterprise applications (Pollock et al., 2003; Somers and Nelson, 2001), there is actually something running to measure, even though it is likely not operated within the enterprise that applies capacity management. If widespread and well-established standard software is utilized, there is a high likelihood that components of the system whose capacity is being managed, are already in production in various environments. Moreover, those components may already produce performance-related log data as part of application performance monitoring (APM) activities (Brunnert et al., 2015; Rabl et al., 2012; Sydor et al., 2010; Gartner, 2017). Such data contain information in terms of performance affecting patterns that need to be extracted and processed in order to transfer the implicit knowledge to other environments. As confirmed in the ITIL service design publication, "[...] it is beneficial to identify similar customers of the product and to gain an understanding of the resource implications from them" (Hunnebeck et al., 2011, p. 174). For this purpose, a digitized capacity management process may incorporate machine learning techniques in order to design a pure black box approach that does not require potentially expensive expert knowledge about the system structure and behavior. This way, performance assessment is enabled already in the design or redesign phase of planned or existing enterprise application environments and can be applied, e.g., as part of server consolidation efforts. The resulting approach serves as another example for the introduced era of digital transformation with the objective to increase efficiency and save personnel expenses.

To account for the aspects outlined above, the data driven capacity management method, presented in this work, is designed for the optimization of large environments of heterogeneous servers and allows for the definition of typical constraints when placing services on available servers. To reflect the QoS, a performance evaluation component integrates machine learning techniques in order to predict response times on the level of business transactions. This is enabled by the cross-environmental nature of the method, which makes it applicable already in the design stage without the need to involve costly test systems and expert knowledge. The method is termed *Performance prediction supported service placement (PPSS)* and is expected to help transforming arguments from an expert- and, therefore, sometimes opinion- or political driven manner to a more solid and less questionable process. The approach complies with a recent Gartner report which identified the improvement of process efficiency as the most expected application field to be addressed by data and analytics, followed by the development of new products and the enhancement of customer experience (Gartner, 2018a) as also intended by the present work.

1.2 Research design

The term design may refer to both the process of design and its output ((Hevner et al., 2004) cited from (Walls et al., 1992)). This section introduces the research design which includes the overall research goal addressed by the thesis at hand and the process of achieving the objective. In business informatics, two common research goals are to reduce costs or to increase quality, usually enabled by the use of information technology. Since the research artifact presented in this thesis was developed in the context of an industry-funded research project, those two objectives hold true as initial impulses for the presented work either. More specifically, the process followed to achieve the objective relies on a hypothesis, which can be expressed as follows:

Hypothesis
The usage of off-the-shelf enterprise applications is dominated by built-in standard business transactions over customized functionality. Therefore, capacity management tasks can be supported by standardized algorithms and performance models that were trained and tested on the basis of monitoring data from various operational environments.

A validation of the hypothesis would enable to achieve the overall research goal. Since machines learn faster than experts (seconds over years), concepts related to big data and machine learning may have the potential to enable new capacity management services. These services enable to leverage economies of scale and to reduce optimization costs while ensuring satisfactory solution quality. As a consequence of the ongoing trend to consume computing power as a service, the capacity planning of large environments from scratch rarely occurs as opposed to the optimization and enlargement of existing environments, which may result from changed or additional business requirements. Such optimizations are usually carried out as part of server consolidation projects which aim at the minimization of total resource capacity while ensuring sufficient performance for any running service at any time. Therefore, the application area of the research artifact is narrowed down to the central problem of service placement within an existing environment; it is known to be a complex task that incorporates a number of challenges (Cherkasova and Rolia, 2006; Müller et al., 2016b). Hence, the research goal of the presented work is defined as follows.

Research goal
The research goal of this thesis is to evaluate the potential of cost reductions for enterprise application consolidation efforts while improving solution applicability. The goal is pursued by applying standardized algorithms and prediction models to the problem of service placement in a way which efficiently supports optimization decisions. Solution applicability is subject to the compliance with given realities of the solution's application area. Therefore, decision support must consider total costs of a solution, resulting from its operations costs on the one hand and compliance with

performance-related service levels and constraints on the other hand.

The strategy to achieve the goal is to increase standardization and reduce man power by centralizing parts of the capacity management process and supporting those parts using black-box techniques that do not require expensive expert knowledge. A successful demonstration of the strategy validates the hypothesis and allows to achieve the research goal. To achieve the objective, a useful artifact is to be designed. In contrast to behavioral science whose goal is truth, this research is conducted by applying the design science paradigm whose goal is utility (Hevner et al., 2004). Therefore, the design science research framework was adapted for this research project as presented in Figure 1.1. Following this methodology, the designed artifact (depicted in the center of Figure 1.1) is a method to solve a problem which includes formal algorithms and machine learning techniques. As pointed out by March and Smith (1995), methods solve problems by translating one model into another model. In this context, models abstract a real world situation (Hevner et al., 2004) using constructs which provide domain-specific vocabulary to describe a problem and its solution space (March and Smith, 1995). The method, presented in this work, aims at translating a model which represents the present design of an application environment into a new model, representing an optimized design of this environment. Constructs, used by the models to represent designs, include capacity providing and capacity consuming entities such as servers and services, their allocation, as well as individual placement constraints. To demonstrate feasibility and suitability to the intended purpose, an instantiation of this method is applied to a real-world problem from the targeted environment. This concerns the service placement problem and involves a number of constraints and domain characteristics which represent business needs that are to be gathered and addressed by the artifact. This way, research relevance is assured. To build and evaluate the artifact rigorously, existing means from the scientific knowledge base are used. Finally, as utility informs theory (Hevner et al., 2004), the artifact itself contributes to the knowledge base. Therefore, a number of research questions (RQ) are raised in the following whose answers help to achieve the research goal and may serve as groundwork for future efforts in the field of capacity management.

The formulation of research questions helps to identify relevant measures for evaluation and reveals contributions that are being made during build and evaluation iterations. Therefore, the process of providing answers contributes to the scientific knowledge base and, finally, produces an evaluated artifact which can be applied in an appropriate environment (Hevner et al., 2004). In the following, each research question is introduced.

The hypothesis presumes that application performance monitoring data from different operational environments can be integrated in order to serve as training data for performance models. The fact that enterprise applications broadly utilize COTS software (Somers and Nelson, 2001; Pollock et al., 2003; Pries-Heje and Dittrich, 2009; Dittrich et al., 2009; Holland and Light, 1999; Hong and Kim, 2002) contributes to this assumption. However, even standard software comprises both standard business transactions and

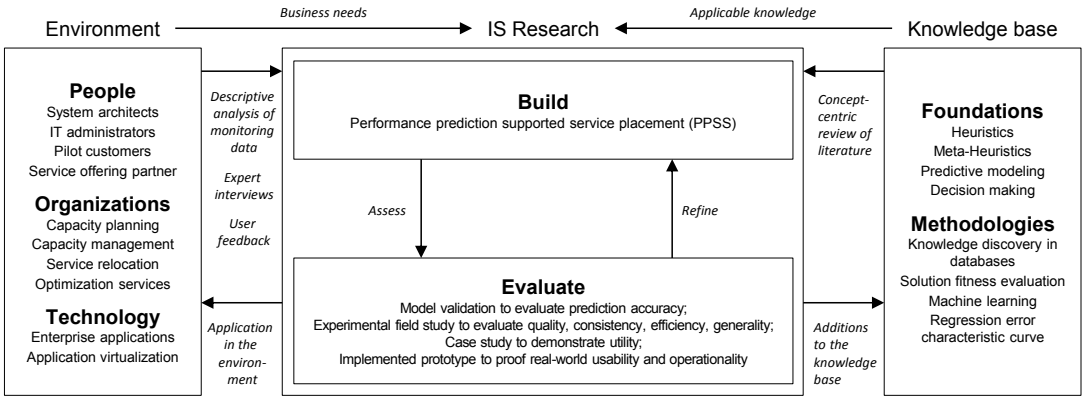

Figure 1.1: Design Science framework used to achieve the research goal (based on Hevner et al. (2004)).

customer-specific transactions. Standardized performance models, in turn, must be related to an entity that is present across various measured instances of the same software. In the domain of enterprise applications, business transactions represent objects which provide functions for different processes within the enterprise such as the collection of information or the interaction with business partners (SAP, 2018a). Hence, to ensure a vast amount of training data, usage patterns of the EA must include business transactions that are available to many instances of the software. This would not hold true for customer-specific transactions that are created to serve specific needs of individual environments. Therefore, a certain utilization level of built-in functionality is required to support the hypothesis. This is investigated using the example of a major COTS EA vendor.

Research question 1
How standardized is a widespread commercial off-the-shelf EA used in practice?

In addition to a sufficient standardization degree, it is important to investigate actual server utilization levels in order to validate the practical relevance of the work. In general terms, the lower utilized the resources, the larger the optimization potential that can be addressed by consolidation approaches.

Research question 2
What are average server utilization levels in environments of widespread commercial off-the-shelf EAs in practice?

EAs are of exceptional significance for business continuity (Müller and Turowski, 2015). Consequently, the application area of the artifact is characterized by various operational constraints in order to guarantee reliable operations according to agreed service levels. Such constraints may be related to, e.g., high availability, information security or performance. It is therefore part of the research to identify and formulate existing constraint types so that the same can be reflected during the optimization process.

Research question 3
Which types of service placement constraints exist and how to formulate them in order to enable effective consideration by optimization algorithms?

Solutions to the service placement problem are obtained by means of optimization algorithms in order to achieve a high degree of automation. The algorithms must consider aforementioned constraints and special characteristics of the problem which result from the application area of EAs. It is to be investigated which type of algorithm addresses the problem most suitably.

Research question 4
What are suitable algorithms to solve multi-dimensional server consolidation problems in order to feasibly balance performance and costs within EA environments?

The hypothesis implies the application of machine learning techniques to predict performance measures within the application area. Therefore, the accuracy of this approach is to be investigated by means of model validation techniques from the data science domain.

Research question 5
How accurately do machine learning techniques predict the performance of a commonly used standard business transaction if they were trained on mass data from different operational environments?

Answers to the research questions are gathered by different means. Real monitoring data, provided by the project's industry partner, serves as one major source of information. The monitoring data was initially transformed into a relational database schema and imported into an in-memory database for subsequent analysis and processing. The schema forms the data layer of an application performance knowledge base that could be queried throughout the entire research project in order to validate assumptions and to gather test data for experiments. Further evidence results from expert interviews and reviews of the scientific literature.

To carry out effective design science research, Hevner et al. (2004) define seven guidelines which they advise to address in a project-specific degree. Below, these guidelines are aligned with the present research project in order to introduce scientific details of the work.

Guideline 1, **"Design as an artifact"**, defines the intended output of the research. In this thesis, a method is designed which can be instantiated by capacity managers in order to identify optimized designs for existing EA environments with respect to costs and performance. The artifact includes algorithm types and model types which are instantiated for the purpose of demonstration.

Guideline 2, **"Problem relevance"**, is investigated both practically and scientifically. The work at hand was conducted as part of an industrial research project with the clear objective to make research results available in the market. Therefore, customers and

experts of the author's industry partner provide business needs which are addressed by the artifact. Furthermore, instantiations of the artifact are offered as services to pilot users who provide feedback with respect to the practical relevance and usefulness. At the time of writing this dissertation, a business model is designed in order to integrate parts of the research output into the service portfolio of the industry partner. On the other hand, the research gap is elaborated by means of the scientific literature.

Guideline 3, **"Design evaluation"**, is demonstrated observational, experimental, and descriptive. Sonnenberg and vom Brocke (2012) point out that evaluation should be conducted continuously throughout the research project in order to assess the progress of the evolving artifact. Therefore, the authors argue for a design-evaluate-construct-evaluate pattern which may expand the well-known build-evaluate pattern. The work at hand reflects this pattern as the design phase of the artifact relies on accurate performance models which must be evaluated ex ante, that is, according to Sonnenberg and vom Brocke (2012), before they are put into use. Additional ex post evaluation activities are carried out after the construction of the artifact. Here, evaluated performance models are applied as part of the artifact in a case study. Typical evaluation criteria for methods include their efficiency, generality, operationality, quality and ease of use (Sonnenberg and vom Brocke, 2012; Hevner et al., 2004). As proposed by Sonnenberg and vom Brocke (2012), evaluation of this work's artifact was carried out on four levels, which are termed EVAL 1 to EVAL 4. Table 1.1 lists the evaluation criteria and applied methods for each evaluation level. The last column of Table 1.1 points to chapters of this work that address the respective activities.

Evaluation level	**Evaluation criteria**	**Evaluation method**	**Chapter**
EVAL 1	Relevance, importance, novelty, (economic) feasibility	Literature review, focus groups, expert interviews, market analysis	State of the art, Design
EVAL 2	Applicability, level of detail	Descriptive analysis, benchmarking	State of the art, Design
EVAL 3	Solution quality, efficiency, consistency, generality, utility	Experimental field study, demonstration in a case study	Evaluation
EVAL 4 (partly)	Ease of use, operationality, fidelity with real world phenomena, impact on artifact environment and user	Productive use by pilot customers in reality, survey, semi-structured interview	Evaluation

Table 1.1: Applied evaluation levels in this work.

Evaluation activities on the level of EVAL 1 are intended to provide certainty about the research relevance and novelty. Therefore, the state of the art is investigated, both

from a scientific and practical point of view. The classification of related artifacts helps to identify a research gap. Certain aspects of the work were evaluated using focus groups of industry experts who ensure practical relevance and importance, e.g., during the assessment of placement constraint types in Section 3.2.3. Given a relevant research gap, activities of EVAL 2 evaluate the applicability of a designed artifact. Here, characteristics of the application area are to be considered as analyzed in Section 2.1.1 (standardization degree of EAs) and Section 2.2.1 (utilization levels of EA servers). On the other hand, design decisions of the artifact must be evaluated before actual construction, e.g., via benchmarking. This includes the validation of models as carried out in Section 3.3.2. As the artifact comprises service placement algorithms and performance prediction techniques, EVAL 3, for the sake of clarity and comprehensibility, is divided into EVAL 3.1 and 3.2. Service placement experiments of EVAL 3.1 belong to the class of experimental evaluation since the constructed artifact is executed in artificial use cases. However, the data, in all investigated cases, comes from real business environments. Therefore, characteristics of a field study are fulfilled either, resulting in an observational evaluation. In that sense, EVAL 3.1 activities are referred to as field experiments. Results of the field experiments are analyzed with respect to consistency, solution quality and efficiency of algorithmic performance. Due to the number of experiments and the variety of used data, generality is evaluated either. While EVAL 3.1 evaluates the artifact in a rather quantitative manner, for EVAL 3.2, a qualitative approach is chosen as it is intended to evaluate the utility of solutions. This may be carried out descriptively and scenario-based by constructing detailed scenarios around the artifact (Hevner et al., 2004). Therefore, the prediction component of the artifact is applied in a case study which was constructed to study the artifact in-depth and to demonstrate utility. Finally, EVAL 4 requires to apply the artifact in a business environment. Therefore, the two main components of the artifact (placement and prediction) are technically implemented as back end engines of the method. Furthermore, the artifact relies on the existence of an application performance knowledge base whose implementation is described in Section 4.4, too. This way, it could be studied how the artifact fits into the technical information system (IS) architecture (Hevner et al., 2004). In addition, a user interface and a feature to import customer-specific data into the application performance monitoring knowledge base (APM-KB) enable the usage of the artifact by a group of pilot customers for their individual capacity management challenges. User feedback was collected and analyzed as part of this ex post evaluation step. Therefore, EVAL 4 is carried out observational and analytical. However, since only a subset of the constructed components was offered to the pilot users, resulting user feedback and drawn conclusions are limited. Therefore, activities on the level of EVAL 4 are referred to as partly EVAL 4 evaluation.

Guideline 4, **"Research contributions"**, is addressed by answering the research questions. As the work at hand describes an interdisciplinary research project, findings contribute to the domains of business informatics, multi-dimensional optimization, and data science.

Guideline 5, **"Research rigor"**, is fulfilled by applying established methods from the field of optimization and knowledge discovery. Consequently, the artifact is built upon existing findings from the scientific knowledge base, including heuristics and metaheuristics as well as prediction model types. Limitations of the method are clearly discussed.

Guideline 6, **"Design as a search process"**, requires to satisfy existing laws in the application area when building the artifact. Due to the study of existing literature and the continual consultation of industry experts, the artifact is aligned with characteristics of the problem environment. Examples include the consideration of identified service placement constraint types and existing server heterogeneity of the environments under study.

Guideline 7, **"Communication of research"**, is accomplished to both the scientific community and practitioners who utilize the artifact. Resulting publications include international conference papers, journal articles and practical guides. The subsequent paragraph summarizes publications being made by the author to communicate the research.

1.3 Publications of the author

The work at hand builds upon previous publications, evaluated by reviewers of international conferences and journal committees. However, this thesis is intended to integrate all parts to a coherent and value-adding artifact. In the following, previous publications are summarized in chronological order and put into context of this work's research goal.

While the problem relevance was unquestioned by the industry partner, a research proposal was published with the intention to receive initial feedback regarding the research relevance from the scientific community (Müller and Turowski, 2015). As part of the compiled review, experts of the addressed field rated relevance with 20 and originality with 18 on a scale from 1 (not relevant respectively poor) to 20 (very relevant respectively very good). Given certainty of the planned artifact's relevance, the data to be analyzed was imported into a relational in-memory database. The import process of the dataset, comprising around 230 million records, was investigated from a scientific perspective in order to improve import performance, resulting in a publication regarding self-configuring data imports for cloud environments (Müller et al., 2015b). In the same year, Bosse et al. (2015) address the evaluation of design alternatives for IT service providers with respect to availability, performance and costs by means of analytical prediction models.

In 2016, two major publications addressed the problem of service placement with the objective to improve energy efficiency and reduce total cost of ownership (Müller et al., 2016b; Müller and Bosse, 2016). In order to validate the relevance measurably, Müller et al. (2016b) identify optimization potential by analyzing mean server utilization levels. A study of related work resulted in a research gap that was addressed by formulating a one-dimensional bin packing problem with the objective to minimize the number of servers. To identify design alternatives, seven algorithms, including heuristics and metaheuristics were developed and evaluated with respect to solution quality. In an experimental setting of a multi-case study, four real EA environments of different sizes were optimized, reducing

the overall server capacity significantly. Subsequent iterations improved the approach as published in (Müller and Bosse, 2016). An analysis of monitoring data revealed that both CPU and memory demand must be considered in the problem formulation, resulting in a dynamic multi-dimensional bin packing problem with two resource dimensions and one time dimension. Eight algorithms were used to solve the problem, representing improved versions of the algorithms presented in (Müller et al., 2016b) and one additional grouping genetic algorithm. For evaluation, historical workload traces from four productively operated data centers were used. A best-fit heuristic that uses a genetic algorithm revealed best solutions, saving up to 53% of existing capacity while effectively reducing the risk of resource overloads.

The energy consumption of IT services was further investigated in (Müller et al., 2016c). The developed procedure model enables to assign monitored energy consumption of hardware components to business transactions so that costs can be allocated to overlaying IT products or organizations. By means of multiple linear regression, energy consumption was predicted on the services layer using workload data from a performed benchmark. Results are intended to attain more granular energy monitoring in order to comply with a product-oriented information management and the ongoing use of cloud offerings in business departments.

After it could be shown how to solve the service placement problem with respect to the minimization of operational costs, an additional research stream focused on the quality of service (QoS). To guarantee that the target design will comply with service level agreements in practice, two fields were further investigated to be incorporated into the problem: Performance prediction and service placement constraints.

In (Müller et al., 2017c), six different prediction techniques were investigated to be applied on application performance monitoring data. Historical workloads from more than 18,000 instances of COTS EAs were used to train and test models which predict the mean response time of selected standard business transactions. Promising results enable to enrich capacity management activities by performance predictions in order to analyze what-if scenarios in the early design phase before actual service placement.

In (Müller et al., 2017a), the results are combined to a concept which enables to apply the adjusted artifact in its intended environment. Therefore, a BPMN process model is introduced which enables to deliver capacity planning as a service to potential consumers. The model is based on an established capacity planning process for web services but was adapted to fit the characteristics of enterprise applications. Furthermore, tasks from knowledge discovery and data science are incorporated to ensure accurate performance predictions. In a scenario-based evaluation, published in (Müller et al., 2017b), the approach is applied to evaluate alternative service placement solution candidates with respect to resulting service performance. On the contrary, (Müller et al., 2017a) demonstrates the usefulness of the developed approach in a capacity planning scenario where a new EA is to be sized in alignment with alternative workload scenarios.

Further important aspects of service quality were investigated by incorporating op-

erational requirements related to, e.g., high availability or information security, into the problem formulation. Such requirements result in a number of service placement constraints which must not be disregarded during the optimization to ensure practical applicability in the domain of EA. In the course of writing this dissertation, a journal article is under review, analyzing the effect of eight placement constraint types on optimization algorithm performance and solution quality (Bosse et al., 2019). Some of the algorithms and experiment results were also used in this thesis during the design of the placement component and its evaluation on level 3.1.

Finally, a number of publications, not directly related to the artifact presented in this work, evolved over the project duration. As the research involved the use and administration of in-memory database technology, corresponding knowledge was shared in the German SAP HANA administration book (Braasch et al., 2016). Furthermore, a paper and a journal article, published in the beginning of the project, present a database migration method with the focus on performance and self-adaptiveness, contributing to the general paradigm of data-driven software operations in accordance to existing capacity limits (Müller et al., 2014, 2015a).

1.4 Thesis structure

The thesis is structured according to the design science research framework. Therefore, the introduction finishes with this paragraph, leading over to the questioning of relevance and a discussion of the state of the art. Based on the results, a research gap is identified which dictates requirements the research artifact must comply with (EVAL 1).

The corresponding requirement engineering process represents the initial activity of the subsequent design chapter. The artifact represents a capacity management supporting method. Consequently, a process model introduces the designed method before diving into the two back end modules of the process. These modules support the service placement and a subsequent performance projection. The designed method is designed to be used as a service and involves two actors: The capacity management provider and the capacity management consumer.

As argued in the previous section, evaluation activities are incorporated into the document structure following the extended research pattern of design-evaluate-construct-evaluate (Sonnenberg and vom Brocke, 2012). Since the overall method's applicability is limited by the accuracy of the performance prediction module, an early ex ante evaluation activity (EVAL 2) was carried out already in the design stage in order to validate performance models. After successful model validation, the same can be released to production and utilized by capacity management consumers. Their perspective is taken in the evaluation chapter. Here, designed models are applied by means of field experiments and a case study with real data, in order to evaluate ex post criteria such as generality, utility and efficiency (EVAL 3). The final evaluation phase describes the technical and prototypical implementation of the artifact into a productive service, offered by the project's industry partner to a group of selected pilot users. By means of a feedback survey and

semi-structured interviews, additional ex post criteria such as operationality, ease of use and the fidelity with real world phenomenon (EVAL 4) could be evaluated. To summarize, the design chapter describes the artifact's build rather from a provider perspective while the evaluation chapter addresses concerns of the consumer perspective.

The final chapter summarizes the findings and scientific contributions. It discusses the limitations of the work and, on this basis, proposes future research directions. The work comprises three appendixes: Appendix A and B provide additional data and illustrations on the evaluation of performance models and solution algorithms. Appendix C shows screenshots of the graphical user interface that was developed to support evaluation activities on the level of EVAL 4.

Each Section ends with a short summary of its main claims. Throughout the whole document, contributions to the research questions are being made. For the sake of convenient reading, these are highlighted in light-gray boxes.

2 State of the art

This section introduces basics that are related to the application area and to techniques utilized by the research artifact. Furthermore, current approaches to address the research problem are investigated and classified. Based on that, a research gap is identified.

2.1 Capacity management

Capacity Management is a process that aims at provisioning a cost-effective amount of resource capacity to IT services that is aligned with current and future business needs. As defined in ITIL, it is carried out in the service design stage but is also involved in all other stages from service strategy to operation and improvement. Starting out with a brief introduction of EA and data center characteristics, relevant to capacity management, this section provides an overview of main activities and challenges when managing IT capacity.

2.1.1 Characteristics of enterprise applications

Various definitions for EAs exist. This work refers to the term EA as defined by Kusic et al., whose definition builds upon Li and Bauer (2005):

> "We broadly define an enterprise application as any software hosted on a server which simultaneously provides services to a large number of users over a computer network. These applications are typically hosted on distributed computing systems comprising heterogeneous and networked servers housed in a physical facility called a data center." (Kusic et al., 2009, p. 2)

As further pointed out by Li and Bauer (2005, p. 1), an EA may comprise "one or more corporate applications, such as financial, accounting, and human resource applications, frequently providing critical business functionalities". Therefore, the authors conclude that business performance is directly depending on EA performance. Grinshpan (2012) adds that EAs support an ever-changing business environment, to which they permanently need to adjust. Due to the exceptional importance of EAs for business continuity (Müller and Turowski, 2015), performance failures result in severe consequences. As analyzed by Tudenhöfner (2011) and Cherkasova et al. (2009), those may include damaged customer relations, lost income, increased maintenance costs, increased hardware costs, delayed project schedules, and project failures. Managing EA performance is a continuous challenge, due to the complex nature of EAs and their physically distributed and co-dependent components. While individual component performance is well manageable,

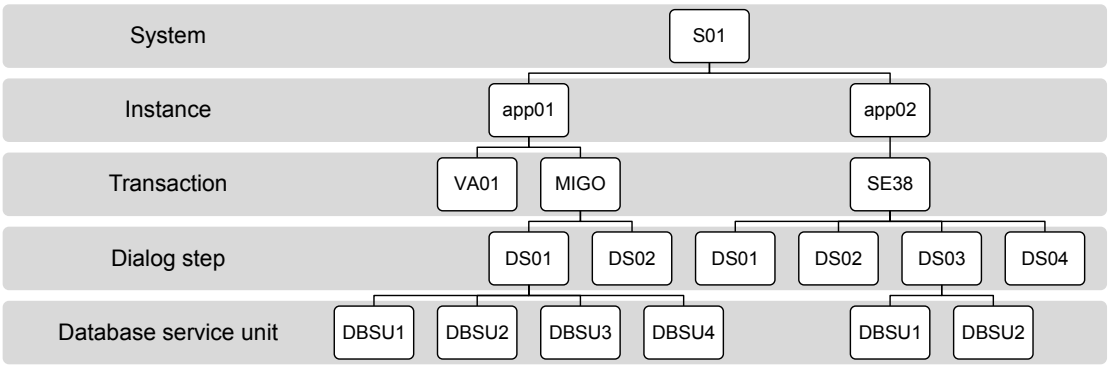

Figure 2.1: Logical EA layers build possible entities for performance measurement.

end-to-end performance from a user perspective involves all logical and physical layers of the EA and, therefore, is hard to predict (Li and Bauer, 2005). From a hardware perspective, those layers typically include a client PC, one or more application servers, network components, and a database server. Accordingly, performance can be measured at different points. Logical entities of an EA may be grouped into the layers illustrated by Figure 2.1 using the example of an enterprise resource planning (ERP) system. Here, the execution of business functions is grouped into transactions which are invoked by users by entering a so called transaction code. After a user logged in to a system, he is dispatched to one of the available application instances and remains there until his user session is terminated. Transactions access data, stored in the database system, via select, insert, update or delete statements. These can be generalized to database service units (Wilhelm, 2001).

The complexity of today's business environments is rarely tackled by internally developed and maintained software, instead, companies implement standardized (out-of-the-box) commercial systems, offered by software vendors such as, e.g., Oracle or SAP, to a large number of customers (Li and Bauer, 2005; Pollock et al., 2003). Sample Identifiers on the right in Figure 2.1 use the example of an EA which utilizes SAP software. Such COTS EAs offer a large number of built-in business transaction types that support standard business processes. Although EA introduction projects seek to use as much standard functionality as possible in order to save costs (Somers and Nelson, 2001), some attributes may require to develop customized business transaction types within the COTS EA.

As Brunnert et al. (2015) point out, performance measurements are dependent on the system and the workload and are not directly transferable to a different system. Therefore, a certain level of similarity with respect to both system type and workload would be needed in order to transfer observations and derive knowledge across different environments, as intended by the research artifact. Consequently, the data basis for model creation must be limited to standard transaction types, possibly used by a large number of customers. For this reason, the hypothesis of this work relies on a widespread distribution of COTS software for EAs and, furthermore, on the assumption that built-in functional-

ity generally dominates the usage pattern over customer-specific code. Only if this holds true, a sufficient amount of training data may be gathered for any cross-environmental performance models.

In a survey, involving 86 organizations who dealt with ERP implementations, Somers and Nelson (2001) identify *minimal customization* as a critical success factor for ERP implementations. The authors cite another survey by Davis and LaMonica (1998), according to which 41% of the fortune 1000 companies re-engineer their business to fit the application while only 5% customize the application to fit their business. Since this survey is relatively old, an analysis of real monitoring data from productively used ERP systems is carried out as part of this work, investigating their actual customization degree. The analysis contributes to the evaluation of applicability of the research artifact, designed in this work.

As the dataset is used in different stages of the present document, it is reasonable to introduce, once in the beginning, a short characterization of the data on which any data-driven design decisions of this thesis are based: Many applications come with integrated software performance monitors (referred to as *software instrumentation*), which produce log information including performance measures whenever certain code is executed (Brunnert et al., 2015). In the domain of SAP systems, those logs are termed statistical records. The total dataset at hand comprises statistical records from more than 18.000 running SAP application instances, each monitored up to three weeks over the last ten years as part of a consultancy service. The total run-time equals more than 700.000 measured hours of system performance whereas 1 exabyte of data was transferred between application and database servers. In total, over 6 billion transaction calls were logged, resulting in more than 16 billion dialog steps (cf. Figure 2.1). While run-time information is suitable to identify workload profiles, the data holds additional information about the landscape topology of each measured environment. More than 16.000 different hardware configurations provide server capacity information, indicating possible performance patterns. Some of the records such as those referring to non-productive systems or incomplete tuples were excluded, depending on the scope of the respective analysis. In the following, the assumption of standard transaction dominance is to be evaluated as addressed by RQ 1. The customization degree of an EA may be computed in two alternative ways:

- **On the basis of transaction calls:** The customization degree equals the percentage of transaction calls which refer to customer-specific transactions.
- **On the basis of CPU time**: The customization degree equals the percentage of CPU time that was spent on processing customer-specific transactions.

Accordingly, the standardization degree equals 100 minus customization degree. Due to the nature of the data model (records are grouped by transaction code and time interval), the transaction-based method disregards redundant calls of the same transaction type in one hour. Therefore, the CPU-based method was designed. Both values can be computed on the level of individual instances, systems or environments. In the domain of SAP

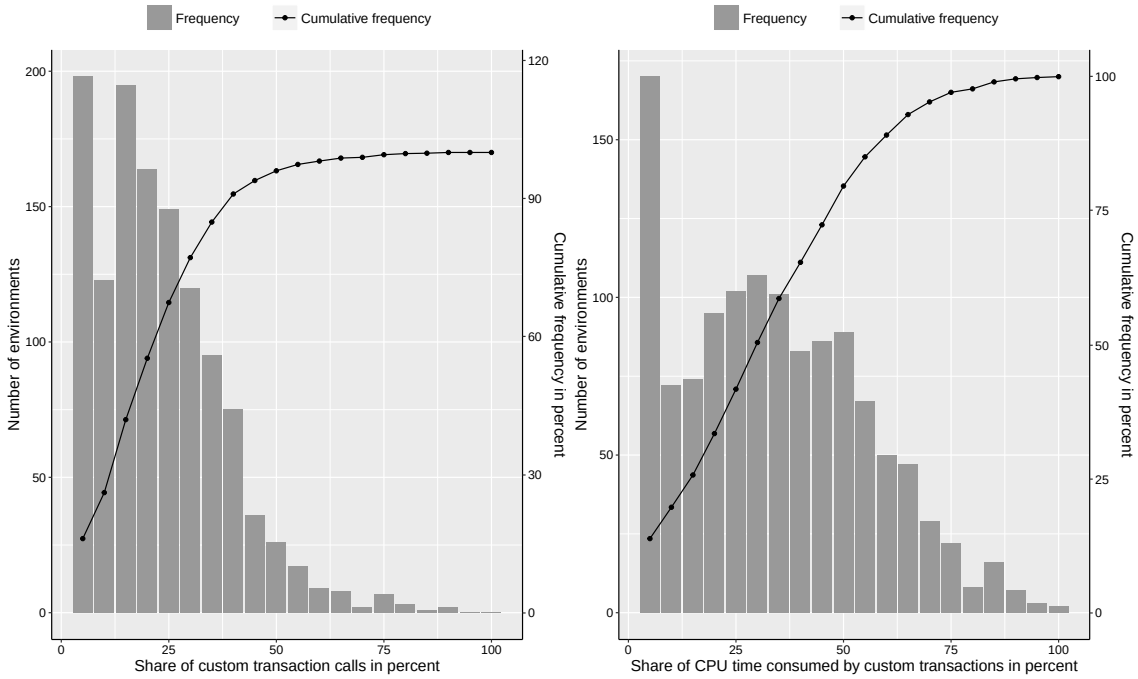

Figure 2.2: Customization degree of investigated EA environments.

systems, customers are encouraged to use a specific namespace when developing own transaction types. Accordingly, codes of customized transactions start with the letters Y or Z. These transaction types are counted when applying the former method. Figure 2.2 shows histograms for the transaction-based and CPU-based customization degree of 1.230 investigated environments whose systems were filtered for productive systems that were measured for more than one week.

With regard to the cumulative frequency, shown in the left of Figure 2.2, about 70% of the investigated environments indicate a customization degree of less than 26% on the basis of transaction calls. Considering CPU time, more than half of the environments show customization degrees below 30% (cf. Figure 2.2 on the right). Mean values account to 20.12% (on the basis of transaction calls) and 31.51% (on the basis of CPU time). It was furthermore analyzed how the size of an environment (represented by the number of running systems) affects the overall customization degree (cf. Figure 2.3). This way, the application area of the artifact may be narrowed down to a certain size of environments.

Most of the data points in Figure 2.3 represent small and medium environments of 1-10 systems with customization degrees below 25%. While the plot does not show clear correlation between the number of systems and the customization degree, it can be stated that customization degrees above 40% are mainly caused by small environments with less than 5 systems. On the contrary, no large environment (> 10 systems) utilizes a customer-specific transaction share of more than 35%. Therefore, the artifact is not limited to but presumably more effective in large environments. This result represents one aspect of the evaluation criteria *applicability* of EVAL 2 (cf. Table 1.1) which is further addressed throughout the thesis. Concrete applicability of the artifact, however, also depends on the

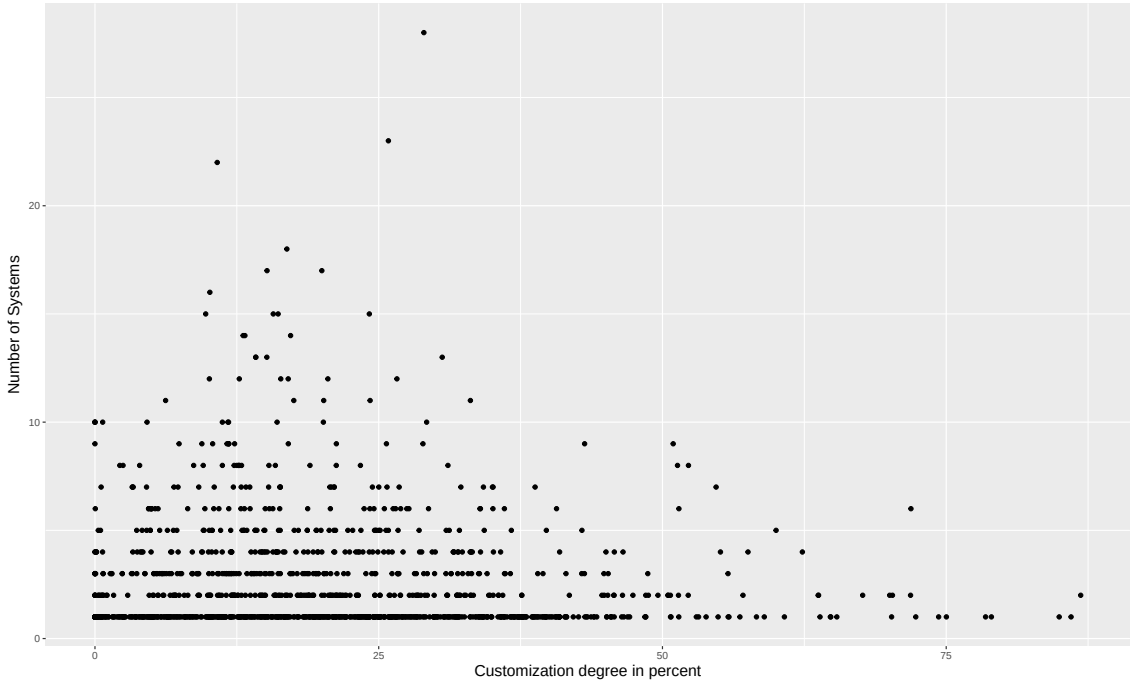

Figure 2.3: Transaction-based customization degree in relation to environment size.

usage degree of transactions under study and must be individually assessed.

Furthermore, it must be stated that customization practice seems to have intensified over the last decades when compared to the survey published in (Davis and LaMonica, 1998). While few statements with respect to customization in EAs could be made, the analysis, however, cannot serve as a snapshot of current system usage patterns as investigated environments may have evolved since the measurement and transaction mix changes over time (Cherkasova et al., 2009; Zhang et al., 2007).

Contribution to research question 1
The vast majority of executed transactions represent standard business transactions. With respect to the frequency of transaction calls, about three quarters of the environments invoke less than 26% custom transactions, resulting in a mean customization degree of 20.12%. In more than half of the investigated environments, custom transactions are accountable to less than 30% of the CPU time. The mean customization degree on the basis of CPU time accounts to 31.51%. Large environments (> 10 systems) do typically not show high customization. In fact, custom transaction shares of more than 35% are solely caused by medium and small environments.

In this work, performance models are built on the layer of transactions (cf. Figure 2.1) as whole instances or systems include too many uncertainties about executed workload due to the existing share of custom transactions. Furthermore, performance from a user perspective is typically measured on a transactional basis and, as Cherkasova et al. (2009) point out, can hardly be generalized to the system layer, since workload from different transactions is barely comparable. For this purpose, the previously determined

customization degree is considered to be small enough as a sufficient amount of training data which refers to standard transaction types, is present.

When measuring performance, two metrics dominate: Response time and throughput. Associated objectives are usually defined by upper and/or lower bounds and related to individual transactions (Brunnert et al., 2015). Objectives related to performance, availability or other non-functional requirements are termed *service level agreements* (SLA) and are usually specified in a contract between the service provider and the service consumer (Boloor et al., 2010a). In this sense, an *IT service* is defined by the ITIL publication concerned with service strategy as a combination of information technology, people, and processes (Cannon et al., 2011, p. 13). More specifically, an EA, as defined in this work, can be made up of one ore more IT services such as application or database instances. The service level agreements must be formulated measurably, precisely and understandably (Appleby et al., 2001; Skene et al., 2010) as part of the *Service level management* (SLM) process. Compensation payments are paid by the provider to the consumer in case the service performs poorly according to the defined SLA (Skene et al., 2010). This is referred to as *violation* or *penalty costs*. The extent of penalty costs, in turn, may depend on the severity of the SLA violation (Gmach et al., 2008). As an example for an SLA related to service availability, a service provider may invoke payments of one dollar for every 15 minutes of downtime, possibly limited to a percentage of usage fees to make sure that penalties do not exceed actual costs of service delivery (Baset, 2012). With respect to performance, SLAs are often percentile-based (Boloor et al., 2010b,a; Gmach et al., 2008). Accordingly, a percentage of requests must be processed within a defined time frame and a penalty is invoked for every percentage point under fulfillment. It is the goal of the capacity management process to ensure that defined SLAs are met at reasonable costs by constantly aligning IT resources with the ever-changing business environment (Hunnebeck et al., 2011; Grinshpan, 2012).

2.1.2 Sub-processes and activities

To manage the complexity of EA capacity, standards and best practices are followed. A widely established collection of standards is provided by ITIL, according to which the process of capacity management ensures the existence of sufficient resource capacity as part of the service design stage. Sufficient, in this case, refers to the performance requirements of every single IT service that is operated within the environment. Therefore, management activities address capacity-related issues that concern both services and resources. When aligning resource supply and demand, it is essential to consider current and future business requirements. According to the service design publication of ITIL (Hunnebeck et al., 2011), capacity management is structured into the following three sub-processes:

- **Business capacity management:** Business drives the underlying IT infrastructure. Therefore, existing business plans and the service portfolio, resulting from the service strategy, can be used to translate data expressed in business terms into levels

of service (LOS). As an example, the expected amount of order line items or sales invoices may be translated into transaction throughput rates or complexity classes of respective business transactions.

- **Service capacity management**: It is the goal of service capacity management to manage, control and predict end-to-end performance of operational IT services. This sub-process ensures that the performance of all services is constantly monitored and resulting data, such as patterns, peaks and troughs in the service resource usage, will be analyzed and reported, e.g., in the form of workload profiles. In order to meet the targeted service performance as defined in the SLAs, cost-effective reactive or proactive measures must be initiated. This often involves domain knowledge from the component capacity management.

- **Component capacity management**: This sub-process is in charge of monitoring all physical components such as servers, processors, network or storage devices with respect to their finite capacity and current utilization levels. Recorded data is to be analyzed and reported either. Activities are implemented in order to guarantee a cost-effective component usage that enables to meet defined service levels, aiming at an optimum alignment of hardware and software resources.

Hence, demand management is carried out mainly in a top down manner that ranges from business processes over service processes to aligned capacity plans. The artifact, designed in this work, involves all sub-processes with a focus on service and component capacity. While ITIL describes a comprehensive overview of activities to carry out, little instruction is provided about concrete instantiation of processes. With respect to capacity planning and management, an example of a plausible process model was proposed by Almeida (2002), with a focus on managing the capacity of web applications using workload models and performance models. The same model was adapted by Menascé et al. (2004) to describe a *Model-based performance engineering methodology.*

The steps are depicted in Figure 2.4. As in (Almeida, 2002), the terms *capacity management* and *capacity planning* are often used synonymously, although the authors of (Brunnert et al., 2015) point out that the capacity of existing environments is being managed while for new services, capacity must be planned. As already stated, the process in Figure 2.4 initially depends on input from business capacity management. The first step (*Understand service architecture*) aims at profound understanding of the environment and architecture, that is topology information. Techniques to gather the required information include user group meetings, audits, interviews and document reviews. The subsequent step is to identify workload profiles for each running service, mainly from transactions logs and performance monitors, here referred to as workload models. Those models must contain information on the workload intensity and the resource demand. The workload forecast bases on organizational and strategic information such as the number of employees or the expected production demand. Almeida (2002) lists possible workload patterns to be random, trend or seasonal. In the domain of EAs, workload usually fol-

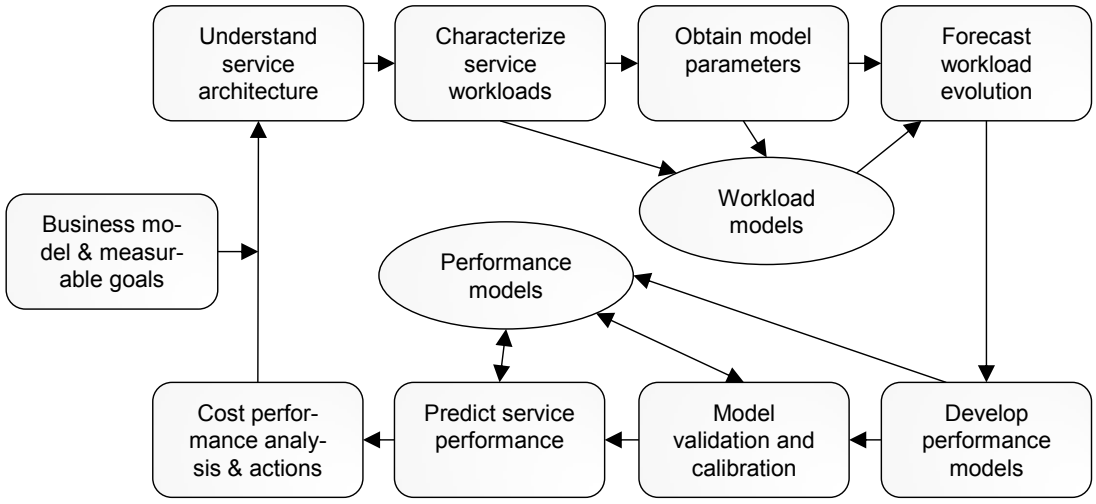

Figure 2.4: Capacity planning process, proposed by Almeida (2002).

lows seasonal patterns on a daily, weekly or monthly basis (Speitkamp and Bichler, 2010). Performance models are developed to analyze so called what-if-scenarios, assuming that service or component characteristics change. Almeida (2002) distinguishes analytical and simulation models. Whereas analytical models utilize formulas to specify component interactions, simulation models reproduce the system behavior in a simulation environment. Both approaches allow for studying effects of parameter changes already in the design stage before actual implementation. Input parameters include (hardware) characteristics of components, workload classes, workload intensity and service demands. With respect to EAs, two common workload classes are batch and dialog jobs. Jobs of both classes can be further classified regarding their complexity as done, e.g., by Menascé et al. (2004) or by Fujitsu (2012) as part of their SAP SystemInspection service. After performance models have been developed, they must be validated (*Model validation and calibration*). Almeida (2002) prescribes to compare predicted values of the model with actual measurement data of the modeled system. Hence, a running instance of the system whose capacity is being managed must have been implemented. In order to validate the model, thresholds of acceptable errors must be defined. In case errors are unacceptable, models are to be calibrated until they match actual performance measurements. By means of a validated model, service performance can be predicted for various scenarios, e.g., changes in workload profiles or server configurations. Again, the observation of a real system serves as a baseline for gathering valid input parameters. The last step (*Cost performance analysis and actions*) addresses the actual goal of the process which is to identify solution candidates in terms of most cost-effective architectures for the environment under study. Therefore, future scenarios are to be analyzed by considering workload profiles, operations costs, and the QoS. Based on the analyzes, actions are recommended in the form of a capacity plan, which is the main output of the capacity management process (Hunnebeck et al., 2011).

2.1.3 Measures and triggers

The individual sub-process goals can be usually achieved by common measures (actions). According to the capacity management process described in the ITIL publication dedicated to service design (Hunnebeck et al., 2011), these measures are formulated as recommendations within the capacity plan. Accordingly, possible recommendations that may lead to a cost-efficient and SLA-reflecting resource usage, can be generally classified as follows:

- **Planning and budgeting of hardware or software upgrades:** By adding or removing resources, the capacity supply can be adapted in order to fit current and future business needs. Such infrastructure changes often cause a subsequent relocation of services in order to optimize resource usage.
- **Cost-effective balancing of services across existing resources**: Depending on current utilization levels, different allocations of services and servers may lead to improved performance or reduced operations costs. Via the identification of complementary workload profiles, allocations can be optimized, likewise resulting in the relocation of services. This is often referred to as workload or server consolidation.

In order to construct a sufficient decision basis, capacity management should involve monitoring patterns of business activity and performance of services as well as utilization of infrastructure components. For this purpose, the monitoring activities carried out during service operation can produce a sufficient basis of data to be analyzed. Based on these data, information can be generated that decision makers may use to decide, e.g., which components to upgrade to which extent so that over-provisioning and bottlenecks are avoided. The capacity plan consolidates these information to appropriate recommendations. Besides documenting current levels of utilization and performance, the capacity plan may additionally include forecasts and assumptions, which should be clearly indicated. Any recommendations made must be quantified with respect to their costs, benefits and impact. Capacity plans are usually produced annually following the fiscal year since it may involve investment recommendations. Aside from this cycle, a quarterly re-issue is conceivable to account for the speed of technological development and occurring business changes. Before the implementation of any measures, their validity must be tested and undergo a structured change management process in order to reduce the risk of affecting current service operation. While the validation of possible measures represents a key aspect of this work, their actual implementation is out of scope. Besides the periodic revision of current capacity and performance, the capacity management process can be triggered by further different events, including

- changes of service composition, e.g. a new service is planned,
- revisions of business plans and strategies, and
- revisions of SLAs or other contracts.

The outcome from the capacity management sub-processes is recommended by ITIL to be stored in a capacity management information system (CMIS). Through this single

source of information, capacity and performance reports and the actual capacity plan, including any forecasts and recommendations, can be made available. In the subsequent section, the CMIS is discussed in more detail. Besides the capacity management process itself, many other processes need to rely on the output:

- Change management to assess the effect of changes on currently available capacity.
- SLM process when new services are implemented to guarantee performance of new service without affecting existing ones.
- Problem management to analyze root causes for incidents of poor performance.
- IT service continuity management to be provided with capacity requirements of the key business processes.

As already stated, in the literature, capacity management is sometimes differentiated from capacity planning, where planning refers to activities required to design and deploy new services, usually involving to identify required hardware components. In ITIL, this activity is referred to as application sizing. On the contrary, in this work, capacity management refers to the optimization of existing services and components that are already in production, e.g., through server consolidation. It is a critical success factor to measure the cost-effectiveness of capacity management activities. According to ITIL, valid metrics to do so may be

- the amount of reduced over-capacity, or
- the accuracy of forecasts with respect to planned expenditures.

These metrics are used when evaluating the artifact, more specifically, as part of EVAL 2 and EVAL 3.1 (cf. Table 1.1).

2.1.4 Information management and reporting

The capacity management process involves monitoring service workloads and transaction usage. Information on resource utilization must be continuously collected via monitors installed on hardware and software components. This is generally carried out through monitoring and control facilities within service operation, also referred to *application performance monitoring*. Aforementioned software instrumentation is a a good example of its practical implementation. Resulting data must be collected, accumulated and stored over a period of time (Hunnebeck et al., 2011). It is the basis for all further management activities and recommendations. Relevant capacity- and performance-related metrics include the following:

- Processor utilization
- Memory utilization
- I/O rates and device utilization
- Database usage
- Concurrent users

- Response time of dialog transactions
- Batch job durations

To classify the data, ITIL distinguishes component-based reports, service-based reports, exception reports as well as predictive and forecast reports. In any case, data should reflect the end-to-end user experience across the complete environment. According to ITIL, different means exist for carrying out application performance monitoring. One or a combination of the following techniques may be used:

- **Software instrumentation:** Special code with the intention to log activities is incorporated into the application. This represents an efficient way to gather actual user response times as done, e.g., in SAP applications in order to create statistical records.
- **Distributed agents:** Software agents are installed at points of interest in order to generate workload and measure resulting performance. Measurement data serves as an indicator for actual user response times.
- **Passive monitoring systems:** External monitoring software measures, e.g., all traffic passing a particular point in the network. Subsequent analysis of the measured data may help to approximate actual user response times.

While intrusion techniques and distributed agents are sometimes classified as gray-box monitoring, on the other hand, passive monitoring systems are referred to as black-box monitoring since they do not have access to application-level logs (Wood et al., 2007). As already stated, measurement data and the resulting capacity plan are to be stored in the CMIS. Since the data also contains valuable information to many other processes (cf. Section 2.1.3), it should be stored centrally in a consolidated way to be available by anyone in charge of capacity- and performance-related decisions. According to ITIL, "only when all of the information is integrated can 'end-to-end' service reports be produced" (Hunnebeck et al., 2011, p. 177).

Based on the collected data, capacity management activities can be defined and carried out. Detailed workload profiles and utilization baselines may be derived for each service or component from the collected data. Baselines are an important reference to benchmark against, so that trends or anomalies can be detected. Component usage may be analyzed on short (24 hours), medium (1-4 weeks) and long term (1 year). Workload profiles are needed to design alternative allocation scenarios or to support effective job scheduling. Finally, as stated earlier, capacity management involves to make predictions and the collected data can be a valuable source to feed performance models.

In the domain of particular COTS EAs, domain-specific metrics such as *SAPS* are commonly used. SAPS stands for *SAP Application Performance Standard* and is particularly interesting if a server's CPU capacity must be mapped to throughput provided in business terms. in this regard, 100 SAPS represent the capacity to either fully process 2,400 SAP transactions or 2,000 order line items per hour (SAP, 2017). Such metrics can

be a useful reference, e.g., when carrying out application sizing activities. The authors of Brunnert et al. (2015) claim a gap between the domains of application performance monitoring (APM) and software performance engineering (SPE). Following their argument, the usage of data collected by software monitors in order to derive valuable knowledge for the design of new services contributes to closing this gap.

2.1.5 Summary

EAs comprise IT services such as application and database instances, typically distributed over a number of heterogeneous servers. Hence, the execution of business transactions involves many co-dependent components on multiple layers which makes performance hard to predict. The vast majority of transactions executed in a COTS EA, represents built-in standard functionality, offered by the software vendor to a large number of customers. Performance is measured on a transactional basis and the fulfillment of related SLAs is crucial to business success. It is the goal of the capacity management process, as defined by ITIL, to balance performance and costs. The process settles the EA software on the layer of service capacity management, which passes requirements from the business down to the component capacity management sub-process. Planning and managing capacity typically utilizes workload profiles and incorporates performance estimates. The outcome of the process is formulated in a capacity plan which introduces recommended measures, e.g., the consolidation of services across existing servers, in order to increase utilization levels and reduce operations costs. The cost-effectiveness of the actions taken can be measured by quantifying the amount of reduced over-capacity or the accuracy of forecasts with respect to planned expenditures. Capacity plans must be aligned with existing workload demands that may be extracted from historical application performance monitoring data. Further relevant metrics, possibly tracked by the application itself, should include the utilization of resources and the response times of dialog transactions. Additional domain-specific metrics such as *SAPS* can help to quantify both service demands and resource capabilities using a system-specific formulation. Any measured data must be stored in a central information system, as other related processes and their actors rely on the information, too.

2.2 Server consolidation

As part of the capacity management process, existing saving potential can be addressed by consolidating orthogonal workloads. Therefore, this section outlines today's data center energy efficiency and reveals average utilization levels of servers. After the general optimization potential was exposed, enabling virtualization techniques are introduced. Next, the problem is formulated and effective means to compute solution candidates are introduced. Solutions must reflect individual constrains of the environment, which are, on a general level, identified in the final subsection.

2.2.1 Server utilization and power consumption

EAs, as defined in Section 2.1.1, serve business requirements and utilize IT equipment which is hosted in a data center and consumes power. While securing the power supply, according to ITIL, is part of facility management in service operation, optimizing consumption and overall efficiency is part of capacity management and thus service design. A datacenter, as defined in the European code of conduct on data centres energy efficiency, "[...] includes all buildings, facilities and rooms which contain enterprise servers, server communication equipment, cooling equipment and power equipment, and provide some form of data service [...]" (JRC, 2008, p. 5). A common classification, made in Bailey et al. (2007) and used by the U.S. Environmental Protection Agency (EPA), distinguishes five types of data centers according to their size: Server closets, server rooms, localized data centers, mid-tier data centers, and enterprise-class data centers. While server closets host only 1-2 servers, enterprise-class data centers are made up by hundreds to thousands of servers (Brown et al., 2007; Johnson and Marker, 2009). As smaller data centers tend to consolidate, large data centers continue to become prevailing (Carr, 2005; Ng et al., 2018). The aforementioned European code of conduct on data centres energy efficiency was initiated in 2008 as a response to the increasing energy consumption of data centers in order to help reducing related environmental and economical impacts as well as threats for secure energy supply. Today, the initiative counts 290 participating data centers (Bertoldi, 2018) which are requested to ensure, among others, the following principles:

- "Data centres are designed so as to minimise energy consumption whilst not impacting business performance.
- Data centre equipment is designed to allow the optimisation of energy efficiency while meeting the operational or service targets anticipated. [...]
- Data centres and their equipment are designed, specified and procured on the basis of optimising the TCO within the requirements for reliability, availability and serviceability. [...]
- Data centres should be designed to minimise the energy used, if any, to remove heat from the facility." (JRC, 2008, p. 13)

Hence, energy consumption is to be minimized cost-effectively without interfacing the business-critical operation (Bertoldi, 2018). In addition, the last item of the list indicates that reduced energy consumption helps to reduce cooling efforts either. In short, energy-efficient operation saves money, since energy costs represent a major share of the data center's TCO. The operation of IT equipment and its cooling facilities account for more than 80% of the total energy consumption of a data center. Among the IT equipment, servers are the greatest energy consumers with a share of more than 75%. Next most significant consumers are storage devices with 10-15% of the IT equipment consumption (Johnson and Marker, 2009). Within a typical server, in turn, the central processing

units (CPU) are the greatest energy consumers (Fan et al., 2007). Enterprise servers are commonly classified according to their prices into volume (<$25,000), mid-range ($25,000-$250,000) and high-end servers (>$250,000). Volume servers, herein, represent by far the majority (about 75% of the total server power consumption) and the fastest growing segment of data center servers (Johnson and Marker, 2009; Shehabi et al., 2016). One characteristic, significantly contributing to the power consumption, is the non-energy-proportionality of volume servers. They consume, in average, about 50% (in some cases up to 70%) of their maximum power usage already at low utilization levels around 10% (Shehabi et al., 2016; Beloglazov and Buyya, 2010a; Barroso and Hölzle, 2007). According to Shehabi et al., this *dynamic range* is expected to drop down to 40% by 2020, indicating that energy-proportional operation is still not in sight. Therefore, idling servers are to be avoided when managing IT capacity. In practice, however, about 30% of all running servers are estimated to be "comatose/zombie servers", thus, showing no CPU, memory, network or other usage activity in a period of 6 months or more (Koomey and Taylor, 2017, p. 2), opening up the potential to optimize the overall utilization. A number of other studies confirm relatively low average server utilization levels in today's data centers of around 10-15% (Beloglazov and Buyya, 2010a; Mi et al., 2010; Gartner, 2011; Müller et al., 2016b) respectively not more than 20% (Van et al., 2009; Kusic et al., 2009).

To verify the conclusions of the mentioned studies in the sub-domain of a COTS EA, monitoring data of more than 13,000 servers which predominantly operate different kinds of SAP systems, was investigated. Since the CPU utilization can be used as an estimator for the server utilization (Pelley et al., 2009; Fan et al., 2007; Splieth et al., 2015), the consumed SAPS of running services were related to the maximum SAPS a server is capable to serve in order to calculate utilization levels. For details on the used data set as well as the metric SAPS, the reader may be referred to Section 2.1.1. The histogram, shown in Figure 2.5, indicates the data distribution for the mean server utilization levels and can be interpreted as follows.

Contribution to research question 2
According to several studies, enterprise servers run at average utilization levels below 20%. With respect to the domain of COTS EA, using the example of SAP software, 75% of the investigated 13,332 servers run with average utilization levels below 12%. More than three quarters of the servers are, according to aforementioned definition, comatose. Consequently, for all investigated servers, the mean utilization, across all measured points in time, accounts to 8.28%.

In practice, however, mean server utilization levels around 65% should be targeted, referring to domain-specific sizing guidelines (Janssen and Marquard, 2007). Low rates can be explained by a common provisioning practice which is based on peak demands and by sizing exercises which are applied *from scratch* to dedicated servers. In many cases, new services are introduced along with new servers (Kusic et al., 2009), resulting in resource allocations best explained by history rather than efficiency.

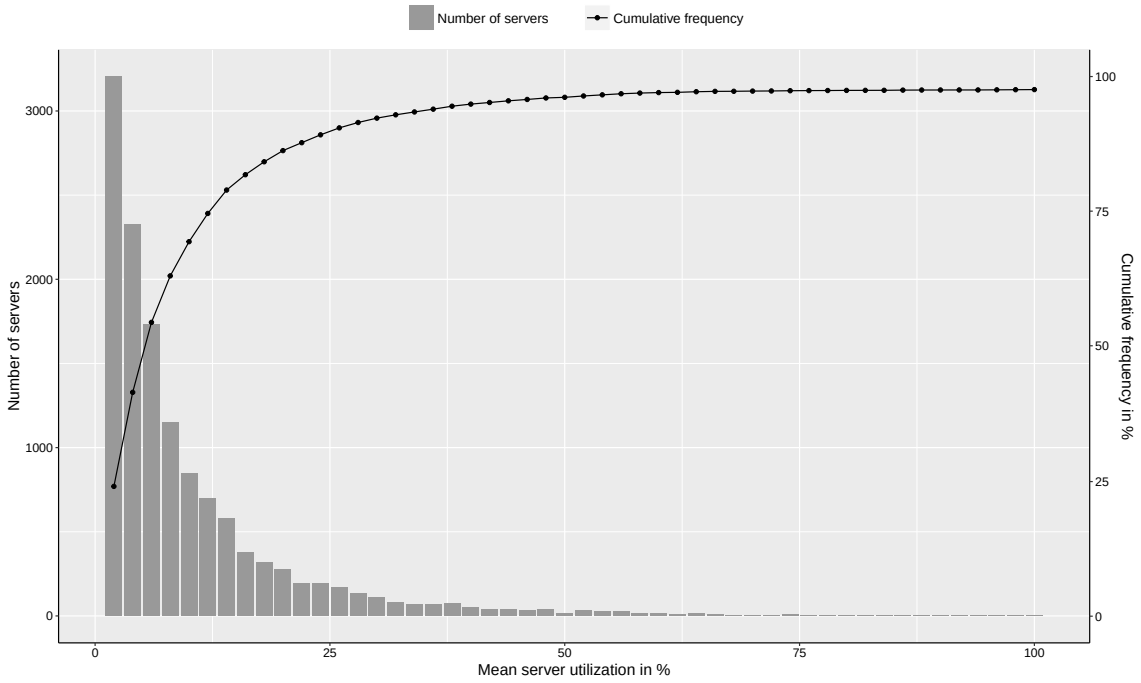

Figure 2.5: Mean server utilization levels across 13,332 investigated servers.

Existing optimization potential can be addressed by means of server consolidation (Gao et al., 2013; Sun et al., 2013) which is a recommended measure to increase energy efficiency in the *2018 Best practice guidelines for the European code of conduct on data centre energy efficiency* (Acton et al., 2018). It is the goal of this process to minimize the total number of required servers by relocating services according to their capacity demands. Due to the non-proportional energy consumption of servers, resulting designs barely effect power consumption of remaining servers but enable to power off a significant share of unused computing resources. According to a Gartner report, among any measures, consolidation has the greatest impact on reducing costs for infrastructure and operation as it affects many components of the cost structure (Gartner, 2009). Reducing the number of servers results in significant cost savings (Petrucci et al., 2011; Xu and Fortes, 2010; Beloglazov and Buyya, 2010b), mainly achieved by energy savings but additionally affecting all related aspects such as cooling (Speitkamp and Bichler, 2010), maintenance and personnel costs (Bobroff et al., 2007).

In June 2016, the *United States data center energy usage report* forecasted the average utilization of active volume servers to be around 15% for on-premises, that is internal, data centers and around 25% for off-premises environments in 2020 (Shehabi et al., 2016). On the contrary, Shehabi et al. outline an alternative, rather positive scenario which relies on the application of server consolidation best practices: In this scenario, utilization levels can be raised from 10-15% to 45% for internal servers, from 20-25% to 55% for servers operated by external service providers, and from 45-50% to 75% for servers of very large and highly efficient data centers. This hypothetical scenario is limited by average upper bounds of utilization which can be observed in well-optimized small and medium size data

centers. Furthermore, it is based on the assumption that 80% of volume servers could be consolidated using virtualization and containerization techniques.

2.2.2 Virtualization techniques

Virtualization is as a key technology to improve energy efficiency (Johnson and Marker, 2009; Koomey, 2011). Consequently, the best practice guidelines for the EU code of conduct on data centre energy efficiency recommend to apply virtualization techniques in order to minimize the amount of hardware that is dedicated to a specific purpose or service (Acton et al., 2018). Likewise, virtualization is mentioned as an important factor when managing IT capacity in the ITIL service design publication. Generally speaking, virtualization refers to the decoupling of software components from hardware components with the main objective to increase flexibility in maintenance and operation. As defined by Wolf and Halter, it is "[...] the act of abstracting the physical boundaries of a technology" (Wolf and Halter, 2005, p. 23). The concept of virtualization was developed by researchers of the Massachusetts Institute of Technology (MIT) and IBM engineers in the 1960's (Daniels, 2009; Susanta and Chiueh, 2005; Bobroff et al., 2007; Goldberg, 1974). As the consolidation of workloads relies on the flexibility provided by virtual environments (Niehorster et al., 2011), a short overview of two main techniques is provided in the following. On a high level, one can differentiate server virtualization and application virtualization.

Server virtualization abstracts hardware to virtual machines so that the operating system (OS) and all installed applications are decoupled from physical components (Bobroff et al., 2007; Müller et al., 2016b). This way, multiple virtual machines (VM or guest system) can be started and operated on one physical server (PM or host system) whose physical resources are allocated to the VMs by a so called hypervisor. Since multiple VMs can be grouped on different hosts, those may be packed efficiently in terms of physical resource utilization. Therefore, the concept of server virtualization enables to optimize resource utilization by consolidating existing servers. Server virtualization is widely used in industry, common products are VMware ESXi or Oracle VirtualBox. A downside of server virtualization results from performance degradations caused by the additional layer of the architecture which the hypervisor introduces (Sahoo et al., 2010). In some cases, relocation of VMs can be carried out in running state, sometimes referred to as live migrations or online migrations. However, live migrations are subject to a number of software and hardware restrictions and further degrade performance by introducing additional network and processing overhead (Ankit et al., 2013; Hyser et al., 2007).

Application virtualization decouples the application from the operating system and, in turn, from the physical components. The operating system is shared across multiple physical servers within the environment that have access to a shared storage layer. Therefore, the OS is made up by common parts (e.g., libraries), mounted to servers in read-only mode and application-specific parts (e.g., database files), mounted in read-write

mode. The concept usually relies on network attached storage (NAS) or a central storage component connected over a storage area network (SAN). Since no additional virtualization layer such as the hypervisor is introduced, applications run at so called *bare metal* and performance degradations are avoided. Further advantage lies in stable operations along with reduced maintenance effort since changes in the operating system image affect all running servers in one sweep. This is, at the same time, the greatest downside since installed OS patches or upgrades must be compatible with all applications that share the same OS image. This fact makes the concept particularly interesting in highly standardized environments where applications are built upon the same technology stack and have similar or equal requirements concerning the OS. This holds also true for SAP EAs which are built upon the SAP Netweaver technology stack (SAP, 2019b). As a drawback, live migrations are not possible and services need to be stopped and started on a different server when consolidating workloads. Therefore, the concept is applicable if workloads follow seasonal patterns and must not be moved frequently in order to balance resource usage. In such cases, allocations must be well-planned according to the workload profiles. Industry examples for application virtualization in the domain of EA, are SAP Adaptive computing (SAP, 2012) and Fujitsu PRIMEFLEX for SAP Landscapes (Fujitsu, 2018). As applications are sometimes referred to as services, Gmach et al. (2008) also refers to this concept as *service virtualization*.

To summarize, the type of employed virtualization technique determines multiple important aspects in managing capacity:

- **The layer on which workload relocations are supported:** Server virtualization enables to move VMs to another PM. This includes the operating system and all applications installed within the VM. In contrast, application virtualization allows for higher level of granularity when computing optimized designs as EAs can be moved individually. This increases the addressable optimization potential as VM (under)utilization barely affects power consumption of the hosting PM while the number of hosted VMs does (Kusic et al., 2009).

- **The state of the EA during relocation:** In some cases, VMs may be migrated in running state, allowing for online optimization of VM allocations. However, if workload profiles can be identified that follow seasonal patterns, online migrations and their associated network overhead may not be desirable and well-planned downtimes for offline relocations can be accepted.

- **Potential performance degradation:** Server virtualization obtains a significant increase of flexibility at the cost of little performance degradations. For EAs, nevertheless, performance is business-critical (cf. Section 2.1.1) and low response times are a major requirement of ERP users (IT-Onlinemagazin, 2015). Therefore, many EAs must be operated on bare metal due to the processing overhead introduced by the hypervisor. In those scenarios, application virtualization can provide the required means of abstraction in order to carry out server consolidation.

The approach presented in this work is not restricted to a certain underlying virtualization technique but is generally applicable in infrastructures that utilize both server and/or application virtualization. In fact, the degrees of freedom when carrying out service relocations depend on the definition of a *service* as the isolated entity that is to be placed. Services can be (a set of) VMs or (a set of) applications that are placed on a target location, that is a physical server. If VMs are to be placed on physical servers, the cloud computing literature refers to the problem of virtual machine placement (VMP) (López-Pires and Báran, 2015; Pires and Barán, 2013). This challenge is sometimes also referred to as the virtual machine mapping problem (Hyser et al., 2007). Since this work is not limited to the placement of VMs, a broader term, *service placement*, is used.

2.2.3 Problem characteristics

Capacity management must ensure an optimum use of hardware and software resources while ensuring compliance with the agreed service levels. According to the ITIL service design publication, this can be achieved by balancing services across existing resources aiming at their most cost-effective usage (Hunnebeck et al., 2011). Since this concept is technically enabled by virtualization techniques (cf. Section 2.2.2), the consolidation of services is highly related to the problem of virtual machine placement (VMP), where VMs are to be located on a given set of PMs (Pires and Barán, 2013). Hereby, VMP aims at improving energy efficiency and resource utilization in cloud environments (Gao et al., 2013). This work, however, is not limited to the placement of VMs. In fact, the entity that is to be placed may be an application or database instance of an EA or an entire VM, depending on the employed virtualization technique. This work therefore refers to the *service placement problem (SPP)* which must be solved in order to consolidate *workloads*.

The mapping process requires two main pieces of information: The capacity of the physical servers and the resource demands of the services (Hyser et al., 2007). All components have a finite capacity that causes performance failures when approached or exceeded (Hunnebeck et al., 2011). Consequently, inadequate placement decisions may lead to either resource overloads and resulting performance degradations or to idling capacity, generating unnecessary operations costs, of which a major portion can be accounted to energy costs (Filani et al., 2008; Orgerie et al., 2014).

Depending on the load fluctuation of running services and the capabilities of the underlying virtualization technique, the problem may be solved online (continuously) or offline (periodically). In the latter case, resource demands of the services can be derived from historical workload data, e.g., stored in a CMIS (cf. Section 2.1.3). If the demand is represented by a single value, e.g., the peak demand of a time interval, the problem is classified as static. In case of dynamic problems, workload profiles represent a service's resource demand over time, reflecting one or multiple resource dimensions such as the CPU or memory consumption. Here, the time dimension may cover maximum hourly, daily or weekly consumption values across a measured time period, representing, e.g., a day of peak values or a week of peak values of the running service. Alternatively, a certain

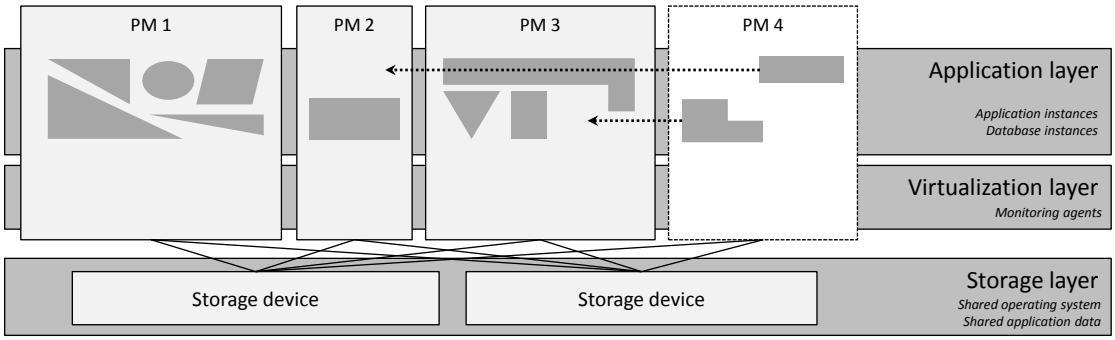

Figure 2.6: Consolidation of application-virtualized services (Müller et al., 2016b).

percentile of those values may be used (Speitkamp and Bichler, 2010). Based on this information, complementary workload profiles are to be identified and placed together on the same server. As the CPU is known to be the greatest energy consumer and, at the same time, the main bottleneck for EAs, it's utilization can be used as the basis to trade-off performance and energy (Splieth et al., 2015; Barroso and Hölzle, 2007; Fan et al., 2007). Figure 2.6 illustrates this process on a high level using a simplified example of four servers (PMs) and eleven services which utilize the concept of application virtualization.

In Figure 2.6, the size of the servers represent their individual CPU capacity while services and their individual load profiles are represented by the differently shaped objects on the application layer. The picture demonstrates the existing fact that some services are more compatible to each other than others, when aiming at the most efficient use of the server capacity. In this example, the services currently running on PM 4 could be moved to PM 2 and PM 3 with the result of an increased mean utilization and a reduced number of running servers. Mathematically, the server consolidation scenario can be formulated as a combinatorial optimization problem, which is closely related to the well-known vector bin packing (VBP) problem, in which the overall size of used bins is to be minimized (Sun et al., 2013). Considering the complexity of today's EA landscapes, the resulting challenge opens up a solution space which makes manual handling of service placements improbable or error-prone (Hyser et al., 2007; Speitkamp and Bichler, 2010). In fact, for a given number of physical servers H and a number of running services I, the number of possible solutions equals H^I (Hyser et al., 2007).

Solution approaches typically formulate one- or multidimensional optimization problems having one or multiple resource dimensions and, optionally, one time dimension. Depending on the optimization criteria, solution strategies may aim at one or more objectives which can be complementary or conflicting. As identified in a literature review by López-Pires and Báran (2015), typical objective functions include the minimization of energy or power consumption, the maximization of energy efficiency, the minimization of network traffic, the maximization of one or more resource's utilization, and thermal management related objectives such as minimizing cooling efforts. While López-Pires and Báran classify problem formulations as mono-objective approaches, multi-objective solved

as mono-objective (MAM) and pure multi-objective (PMO) approaches, Helbig and Engelbrecht (2013) distinguish between single-objective (one objective), multi-objective (up to 3 objectives), and many-objective (more than 3 objectives) approaches.

As server consolidation problems are known to be NP-hard (Bichler et al., 2006), exact solution approaches are not efficient enough for practical use since solutions cannot be found in polynomial time (Lewis, 2009). Instead, solution candidates are usually calculated using mathematical programming, heuristics or metaheuristics (Jennings and Stadler, 2015; Liu et al., 2008; Bosse, 2016; Soltani, 2014). An overview of instantiated approaches from the literature along with a discussion on their advantages and disadvantages is provided in Section 2.4. Furthermore, additional classification attributes are derived from related artifacts in this section. The deployment of computed solutions is out of scope in this work, but may be technically enabled by various virtualization concepts (cf. Section 2.2.2).

2.2.4 Approximation algorithms

Server consolidation is a challenging task that consumes significant time and other resources (Gartner, 2009). In general, the addressed class of optimization problem occurs in many fields of computer science, especially if it is required to make any kind of assignments, e.g., for scheduling and load balancing (Lewis, 2009). Approximation algorithms find appropriate solutions quickly and do not solve the problem exactly. As outlined in the previous section, heuristics and metaheuristics are very efficient techniques to solve the SPP; they have been applied successfully to bin packing problems in related approaches and represent the commonly used state of the art. In fact, more than half of the approaches reviewed by López-Pires and Báran utilize heuristics while 15.5% of the solution techniques involve metaheuristics. Within the class of heuristics, variants of the popular best-fit and first-fit algorithms represent the majority of established techniques. On the other hand, implementations of a genetic algorithm (GA) are among the most used techniques in the class of metaheuristics. Mohamadi Bahram Abadi et al. (2018) confirm that solution techniques for server consolidation are dominated by heuristics and metaheuristics. Therefore, as the most popular representatives in the problem domain, the first- respectively best-fit heuristics as well as the genetic algorithm are briefly introduced in the following.

First- and best-fit decreasing algorithms

Both first- and best-fit decreasing algorithms (FFD and BFD), e.g., as used by Stillwell et al. (2010), rely on the sorting of services according to their resource demands. As services with high resource demands are supposed to be placed first, the list of services is processed in descending order. The next service is then allocated to the first server which provides sufficient remaining capacity. This procedure is depicted using the pseudo code in Algorithm 1.

Algorithm 1 First-Fit-Decreasing

```
1: procedure FFD(T, H, I)
2:     a ← ∅
3:     I_sorted ←SortDescendingByResourceDemand(I)
4:     for i ∈ I_sorted do
5:         for h ∈ H do
6:             fit ← true
7:             I_h ← {i ∈ I : a(i) = h}
8:             for r ∈ R, t ∈ T do
9:                 if Σ_{i_h ∈ I_h} i_h(r,t) + i(r,t) > h(r) then
10:                    fit ← false
11:                end if
12:            end for
13:            if fit = true then
14:                a(i) = h
15:                BREAK
16:            end if
17:        end for
18:    end for
19:    return a
20: end procedure
```

In contrast, the BFD does not rely on an arbitrary order of servers. Here, also the list of servers is sorted before new allocations are made. Since servers with relatively low residual capacity are to be preferred when placing the next service, the BFD sorts servers in ascending order according to their remaining capacity. This procedure efficiently increases load density especially in environments with homogeneous servers (i.e., servers of equal capacity) as services are preferably placed on servers already in use. Algorithm 2 depicts the procedure.

In the case of dynamic problems (i.e., a problem formulation that incorporates varying workload over time) and in case of multiple resource dimensions, the set of different demand values must be resolved for each instance in order to sort the list of services according to their resource demands. According to (Stillwell et al., 2010), the time dimension may be transformed to either a single sum of all values or the peak demand across all values. The resulting algorithms are referred to as FFDmax and FFDsum respectively BFDmax and BFDsum. Those heuristics were effectively applied to both single- and multidimensional server consolidation problems (Stillwell et al., 2010; Bichler et al., 2006). In contrast to the described strategies, an increasing ordering of items was identified to be unsuitable for most bin packing problems (Stillwell et al., 2010). However, a main disadvantage of heuristics lies in their limited flexibility with respect to varying problem formulations. Individual constraints which usually limit the degrees of freedom when placing services cannot be considered by any of the introduced heuristics without major adaptations.

Algorithm 2 Best-Fit-Decreasing

```
1: procedure BFD(T, H, I)
2:     a ← ∅
3:     I_sorted ←SortDescendingByResourceDemand(I)
4:     for i ∈ I_sorted do
5:         H_sorted ←SortAscendingByRemainingCapacity(H,a)
6:         for h ∈ H_sorted do
7:             fit ← true
8:             I_h ← {i ∈ I : a(i) = h}
9:             for r ∈ R, t ∈ T do
10:                if Σ_{i_h ∈ I_h} i_h(r,t) + i(r,t) > h(r) then
11:                    fit ← false
12:                end if
13:            end for
14:            if fit = true then
15:                a(i) = h
16:                BREAK
17:            end if
18:        end for
19:    end for
20:    return a
21: end procedure
```

Genetic algorithms

Genetic algorithms belong to the class of metaheuristics and represent a popular technique for solving optimization problems (Bäck, 1996), e.g., in the domain of server consolidation (Adamuthe et al., 2013; Xu and Fortes, 2010; Rolia et al., 2003; Stillwell et al., 2010) or when it comes to scheduling problems (Brezulianu et al., 2009; Sapru et al., 2010; Page and Naughton, 2005). Genetic algorithms are designed to proceed according to the evolutionary process of natural selection which leads to the *survival of the fittest*. Therefore, an initial, randomly chosen set of solution candidates (in this context, referred to as *individuals*) forms a population which evolves over a number of generations. In order to direct the algorithm in exploring the search space, genetic operators are applied to the individuals. Those operators are inspired by biological operations; more specifically, individuals may be *recombined*, *mutated*, and *selected*. To support the selection process, a function evaluates individuals regarding their solution quality (in this context, referred to as *fitness*). This function is termed *fitness function*. The pseudo code in Algorithm 3 describes the general process as also used in Bosse (2016).

Recombination combines at least two parent individuals in order to create λ new child individuals. Parent individuals are typically chosen to be recombined with a fitness-proportional probability. The resulting subset of individuals (*offspring*) may fully replace the parent population as depicted in Algorithm 3 or complement it. The former approach is termed *comma-selection* and the latter is referred to as *plus-selection*.

Some individuals of the population are further selected to be modified with a certain

Algorithm 3 Genetic Algorithm

```
1: procedure GA(μ, λ, maxGen)
2:     gen ← 0
3:     pop ← INITIALIZE(μ)
4:     EVALUATE(pop,0)
5:     while gen < maxGen do
6:         pop ← pop ∪ RECOMBINE(pop, λ)
7:         pop ← MUTATE(pop)
8:         EVALUATE(pop,gen)
9:         pop ← SELECT(pop, μ)
10:        gen ← gen + 1
11:    end while
12:    return pop
13: end procedure
```

mutation probability p_{mut}. In contrast to recombination, mutation changes solution candidates individually and, therefore, has little effect on the search direction, thus, depicting some form of neighborhood search.

Finally, all individuals of the resulting subset are evaluated with respect to their fitness. Based on this evaluation, most fittest individuals are selected to build the population (with the size *pop*, also called μ) of the next generation. To implement the selection process, several approaches exist in the literature (Kruse et al., 2016). A very simple form is the *rank based selection* where candidates are sorted in descending order and the top n candidates are selected. Here, most fittest solutions always dominate the selection process which entails the risk of converging to a local optimum without exploring the full search space. In contrast, *tournament selection* uses a rang-proportional selection. Depending on the tournament size t, a number of individuals are selected and sorted by fitness in descending order. In this tournament, each individual is selected with a probability p_t proportional to its rang. The procedure is repeated until *pop* is reached. Therefore, selection is not directly fitness-proportional and the problem of dominance is coped to some extent. Selection pressure can be controlled over the tournament size (Miller et al., 1995).

In contrast to the rang-proportional probability of selection, a fitness-proportional probability is used by the *roulette wheel selection*. For each individual, therefore, it is computed the relative fitness, that is the share of fitness with respect to the accumulated fitness of all individuals (Gerdes et al., 2013). Using this information, a pie chart could be built, similar to the one on the left in Figure 2.7 with, in this example, eight individuals s per population. The position of the pointer is defined randomly. Yet individuals of greatest relative fitness are pointed at with highest probability. Those individuals are selected for the next population and the process is repeated until *pop* is reached. Although there is no guarantee for constantly selecting individuals of greatest fitness, the problem of dominance still exists. A variation of the roulette wheel selection, the *stochastic universal sampling* approach, introduces a mechanism that ensures to additionally select individuals of lower fitness. Here, multiple pointers are used as illustrated in Figure 2.7; in fact, the

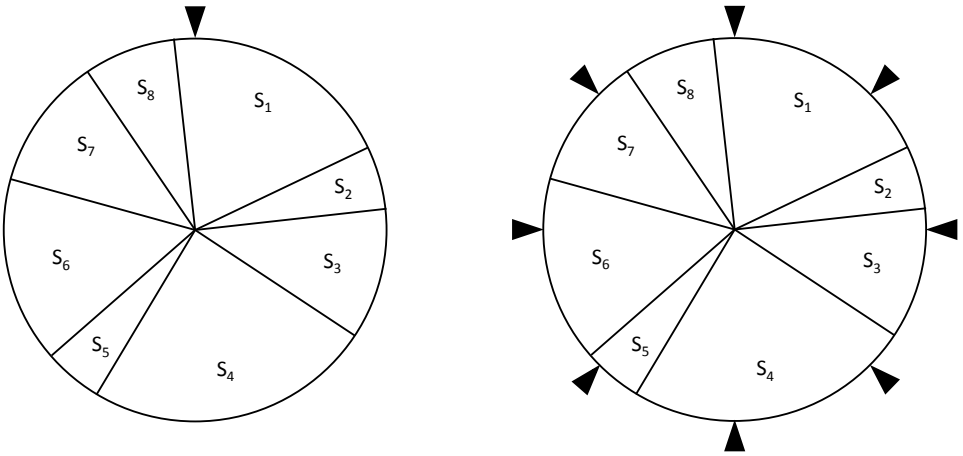

Figure 2.7: Roulette wheel selection (left) and stochastic universal sampling (right), based on Kruse et al. (2016).

number of pointers is determined by *pop*. The pointers are distributed with equal distance around the wheel while the starting position is chosen randomly. All individuals pointed at are selected to build the new population. This usually includes superior individuals but additionally considers individuals of lower relative fitness so that diversity is increased and the search space is efficiently exploited (Baker, 1987). Finally, to ensure that most fittest individuals are not damaged by genetic operators such as mutation, those may be kept and passed over to the next generation without modification. This strategy introduces a so called *elitism*. However, individuals which build the elite can still additionally be selected for genetic operators and, this way, be further improved (Kruse et al., 2016). The genetic algorithm terminates after a defined number of iterations ($maxGen$) or when achieving a defined fitness objective. To summarize, the following numerical parameters affect the behavior of the GA:

- λ: The total number of children produced during recombination
- $maxGen$: The number of generations the GA iterates over
- μ (pop): The number of individuals in every generation
- p_{mut}: The probability of mutation for each individual

As pointed out by Stillwell et al. (2010) and confirmed by the review of López-Pires and Báran (2015), dynamic resource allocation problems are predominantly solved by simple heuristics due to their complexity, even if only a single resource dimension is considered. However, if the optimization potential is addressed insufficiently, solutions were proposed that may not legitimate the cost overhead introduced by capacity management efforts. Therefore, it can be reasonable to compare different algorithms with respect to objective values as proposed by Mills et al. (2011).

2.2.5 Placement constraints

In practice, various dependencies and existing requirements related to, e.g., compatibility, fault tolerance, licensing, or security, limit degrees of freedom in service placement and, therefore, reduce the addressable optimization potential (Shaw, 2004; Hyser et al., 2007; Dang and Hermenier, 2013). As an example, a service may have to be located on a certain type of hardware to ensure compatibility or must not be co-located with its backup service to ensure high availability (HA) in case of hardware failures. Another example of Hyser et al. (2007) draws attention to the increasing use of application service providers (ASP): It may be specified in the SLA of an EA that certain services of a customer must never run together with services of a competitor on the same PM or in the same network. Such limitations are inherent to many practical problems since they usually represent sub-components of a larger environment with various dependencies (Shaw, 2004). With respect to the VMP, Dang and Hermenier (2013) define a placement constraint as follows:

> "A VM placement constraint restricts the placement of a VM or a set of VMs. [...] In a virtualized datacenter, customers express their SLAs through placement constraints. A VM placement algorithm is then in charge of placing the VMs on the nodes according to the constraints stated in the customer SLAs." (Dang and Hermenier, 2013, p. 9, 31)

When replacing the term "VM" by the broader term "service", above definition fits the perception of this work well. Therefore, to fulfill SLAs and avoid violation penalties, individual placement constraints must be incorporated into the respective instantiation of the SPP. Contributing to research question 3, it is the goal of this subsection, to identify existing placement constraints from the scientific literature. The consolidated result (cf. Table 2.1) serves as a basis for additional practical evaluation of relevance together with industry experts and to align the artifact's design. Parts of this study on existing placement constraints was previously published in a Master thesis (Akhras, 2017) supervised by the author of this work.

Although placement constraints practically reduce the search space, additional complexity is added to the problem formulation. Many publications address the problem of consolidating workloads but only few consider placement constraints and even fewer actually implement means to reflect a set of constraints in conducted experiments. In a literature review on the VMP, López-Pires and Báran (2015) identify three types of constraints which were proposed for the problem formulation: *anti-location*, *anti-colocation* and *resources constraints*. The latter can be, e.g., a defined maximum utilization for a specific resource dimension as also proposed in (Pires and Barán, 2013) as a *resource capacity constraint*. Anti-location prevents a service to be placed on a specific server, e.g., due to known incompatibility issues with certain hardware components. In contrast, anti-colocation refers to the restriction of not allowing two or more services to be placed together on the same server. In other works, this is called *isolation* (Calcavecchia et al., 2012) or *separation* (Speitkamp and Bichler, 2010). This constraint type is driven by security or performance concerns.

Speitkamp and Bichler (2010) additionally propose the following four constraint types, where the first two can be seen as representatives of the mentioned anti-location and anti-colocation constraint types: (1) *Preassignment*: a service must be located on a specific server. This is referred to by Pires and Barán (2013) as *unique placement constraint*. (2) *Combination*: two or more services must be co-located on the same server, e.g., to support inter-service communication. (3) *Maximum number of services*: An upper bound for the number of services that must not be exceeded on a specific server, e.g., in order to reduce administration effort in the event of a server failure. (4) *Maximum number of re-allocations*: Particularly for online consolidation approaches, it is necessary to reduce the migration overhead which could be achieved with a defined limit of re-allocations. However, it must be noted that the authors of (Speitkamp and Bichler, 2010) propose the mentioned constraint set but did not incorporate the same into their solution algorithms. Instead, it is assumed that placement constraints would have an effect on solution quality and computing time.

Dang and Hermenier (2013) propose a *location constraint* which equals the preassignment of Speitkamp and Bichler (2010). In addition, the authors differentiate *discrete constraints* and *continuous constraints*. While the former refer to the final design, the latter are intended to prevent SLA violations during the reconfiguration phase of online approaches. The authors of Pires and Barán (2013) add a *service level agreement provision constraint*, which defines whether a service is critical or not. While a classification of services with respect to their importance seems to be beneficial, e.g., in order to prioritize demands, a single Boolean parameter may, however, not satisfy measurable requirements which result from given SLAs.

Jammal et al. (2015) propose a placement solution with a strong focus on high availability of components. This is achieved through the satisfaction of the following four constraint types: A (1) *network delay* constraint defines whether communicating components should be hosted on the same server, rack, data center or within the same cloud offering. While the levels of granularity are reasonable, this constraint type seems to be highly related to the co-location constraint where the location may refer to the proposed levels. Furthermore, a (2) *redundancy constraint* is defined for services that must be placed separately. This requirement, in turn, is identical to the anti-colocation constraint. A subject attainable by the location-constraint is outsourced to a special (3) *failure rate constraint*, requiring to find a server with maximum mean time to failure (MTTF) and minimum mean time to recovery (MTTR). Finally, the proposed (4) *capacity constraint* ensures to avoid overflows. It is sometimes referred to as *resource constraint* (Yusoh and Tang, 2012; Bin et al., 2011) and, in most problem formulations, represents the main restriction.

Aforementioned anti-loaction and anti-colocation constraints are also proposed by the authors of (Bin et al., 2011), whose main objective is to guaranty levels of HA. Therefore, an *availability constraint* was designed which guarantees that VMs always have a secondary PM where they can be moved to in case the primary PM fails. The resulting state must

Constraint type	**Alternative labels**	**Drivers**	**Literature examples**
Resource constraint	Capacity constraint	Performance, availability	(Jammal et al., 2015; Yusoh and Tang, 2012; Bin et al., 2011)
Specific location constraint	Preassignment, unique placement constraint, failure rate constraint, availability constraint, among	Compatibility, licensing	(Jammal et al., 2015; Speitkamp and Bichler, 2010; Bin et al., 2011; Hermenier et al., 2013)
Anti-location constraint	Ban	Incompatibility	(Jammal et al., 2015; Yusoh and Tang, 2012; Bin et al., 2011; Hermenier et al., 2013)
Colocation constraint	Network delay constraint, combination constraint, security placement constraint, gather	Performance, intercommunication, licensing, security	(Jammal et al., 2015; Speitkamp and Bichler, 2010; Shi et al., 2012; Hermenier et al., 2013)
Anti-colocation constraint	Redundancy constraint, separation constraint, isolation constraint, spread	Redundancy, fault tolerance, performance, security	(Jammal et al., 2015; Speitkamp and Bichler, 2010; Yusoh and Tang, 2012; Bin et al., 2011; Calcavecchia et al., 2012; Dang and Hermenier, 2013; Shi et al., 2012; Hermenier et al., 2013)
Maximum number of services constraint	N/A	Administration effort, business continuity	(Speitkamp and Bichler, 2010)
No neighbors constraint	lonely	Performance, administration effort, isolation, security	(Hermenier et al., 2013)
Maximum utilization constraint	Resource capacity constraint, preserve	Performance, availability, optimization degree, risk attitude	(Pires and Barán, 2013)
Extra resource constraint	Preserve	Performance, availability, risk attitude	(Hermenier et al., 2013)

Table 2.1: Placement constraint types as derived from the literature.

still fulfill all other constraints which were defined for the original placement problem.

The most comprehensive portfolio of placement constraints was found in (Hermenier et al., 2013). Their proposed VM placement solution supports the following constraint types: Besides the well-known anti-location, co-location, anti-colocation and location constraints (here referred to as *Ban*, *Gather*, *Spread* and *Among*), seven additional constraint types are of interest: *Preserve* supports to define a minimum amount of resource demand for a VM. Analogue, *Oversubscription* reduces the demands of a VM in percent. *Offline* allows to ignore a specified server when placing VMs, since the same must be offline in the optimized design. *Capacity* is designed to limit the maximum number of VMs. And *Lonely* isolates a specified (set of) VM(s) from all remaining VMs. Finally, two of the proposed constraint types are only relevant for online approaches: *NoIdles* turns off servers in case they do not host a VM and *Quarantine* creates a zone in which no VMs are allowed to be migrated in case servers were compromised.

Table 2.1 consolidates the identified constraint types. While the first column defines labels for each constraint type as used in the present work, the second column provides alternative labels under which the constraint types may be found in the provided literature examples. The listed drivers describe the general motivation and reasons to incorporate respective constraints.

Contribution to research question 3
Consolidation approaches, found in the scientific literature, mention typical placement constraints as listed in Table 2.1. However, most related artifacts consider well-selected constraints and non of the studied approaches is capable to support the entire set of identified constraint types. Furthermore, the effect of constraints on the performance of implemented algorithms and on solution quality was not investigated quantitatively. An additional contribution to RQ 3, carried out as part of the artifacts design, complements the set of constraint types from a practical perspective.

With particular regard to the extensive use of cloud offerings, capacity management affects a growing amount of services and servers operated in a decreasing number of environments. According to Gartner analysts, 90% of Infrastructure as a service (IaaS) providers that existed in April 2017, will have to pass over their business to the prevailing offerings Amazon AWS or Microsoft Azure which host many times more computing power then all other providers. With the ongoing centralization of computing resources, infrastructure is becoming increasingly complex and operational constraints gain more importance than ever before (Ng et al., 2018). Consequently, constraint fulfillment must be an integral part of the solution fitness evaluation in order to ensure practical applicability.

2.2.6 Summary

Du to the non-energy-proportionality of servers, even idle resources consume significant amounts of energy. In practice, server utilization levels typically vary between 10 and 15% which also holds true for the domain of COTS EA. The consolidation of workloads

on a reduced number of servers helps to improve average utilization levels and, therefore, represents a widely acknowledged mean to improve energy efficiency of data centers and to save costs. The deployment of optimized designs is technically enabled by the concept of virtualization, which abstracts running services from physical servers. Those services are to be placed with the objective to minimize required resource capacity. To solve the service placement problem in accordance to available server capacity, orthogonal workload profiles are to be identified. Depending on the state of the services during the relocation phase, one may distinguish between online and offline approaches. In contrast to static approaches, dynamic approaches involve a time dimension that goes beyond a single value. In addition, one or more resource dimensions form the problem, which is known to be NP-hard. Simple heuristics and metaheuristics dominate the portfolio of applied solution algorithms. So called placement constraints are only selectively considered in related approaches although known to be mandatory for consolidation efforts that are aligned with given SLAs. This fact limits practical applicability of proposed approaches. Furthermore, the effect of placement constraints on the algorithm performance was not investigated in the studied literature.

2.3 Performance projection

Predicting the behavior of IT services under a given workload is a main challenge of the capacity management process (Hunnebeck et al., 2011). Therefore, this section introduces common means of performance prediction for EAs with a special focus on machine learning methods. A classification scheme is developed with the intention to compare relevant selection criteria for modeling techniques. Most related prediction approaches to the one designed as part of this work are classified accordingly and compared to a new approach which aims at combining advantages of existing techniques.

2.3.1 Projection objectives and strategies

As carved out in Section 2.1.1, EAs are required to answer timely and performance failures lead to business failures. Therefore, the anticipation of performance characteristics prior to the actual task execution is at the heart of several important decisions in the field of workload management, system sizing and capacity planning (Ganapathi et al., 2009). In particular, anticipations are valuable to schedule jobs, to estimate wait times in queued systems, and to analyze what-if scenarios (Matsunaga and Fortes, 2010). According to ITIL, examples of such what-if scenarios may reflect hardware and/or workload changes: “What if the throughput of service A doubles? What if service B is moved from the current server onto a new server? What will be the effect on the response times of the two services?” (Hunnebeck et al., 2011, p. 173) Especially the latter example points at aforementioned server consolidation exercises. While energy consumption is reduced, consolidated designs may imply the risk of performance degradations, if servers are overloaded (Beloglazov et al., 2012; Jin et al., 2013). In order to not violate given SLAs and avoid

the payment of penalty costs, it is essential to evaluate planned designs with respect to the resulting performance before actual deployment (Chen et al., 2002).

Before deepening the specific problem of EA performance evaluation, general distinctions are to be defined. With respect to the anticipation of future states, according to (Bray and von Storch, 2009), one can distinguish predictions, projections and forecasts, as originating from climate science:

- Prediction: Anticipation of a target value under current conditions, e.g., through extrapolation.
- Projection: Anticipation of a target value under defined conditions in order to answer what-if questions.
- Forecast: Anticipation of a time series in the future, considering the initial state.

Thus, in the domain of capacity management, predictions are helpful, e.g., when the load of a managed system is to be anticipated. In this respect, particularly online approaches may benefit from prediction capabilities. On the other hand, if design alternatives are to be evaluated, performance projections are carried out, e.g., in order to compute response times which result from a solution candidate that incorporates a number of assumptions. Prediction, however, is also used as a general term for any kind of the above listed anticipations.

A major challenge when projecting performance of EAs lies in the complexity of their component-based architecture. While all components of an EA may be implemented using different technologies (e.g., different database management systems), they are tightly coupled and cannot be assessed individually. Understanding the eventual performance of an EA before its deployment is therefore challenging (Chen et al., 2002; Williams and Smith, 1998; Chen et al., 2005). In this context, performance refers to the throughput or the response time associated with a service or a component. Research mainly applies monitoring based systems and performance models in order to assess the performance of an EA prior to the operations phase (Gmach et al., 2008; Menascé, 2003). Accordingly, both Becker et al. (2006) and Brunnert et al. (2015) distinguish the following two quantitative strategies on a high level:

- **Measurement-based performance evaluation** refers to setting up a real system or its prototype and running scripts that simulate user interaction with the system in order to create load, also referred to as benchmarking. A prominent benchmark in the domain of SAP systems is the sales and distribution (SD) benchmark which simulates a defined number of end-users who concurrently log in, execute a sequence of SD transactions and log off (SAP, 2019a). During this process, relevant metrics must be measured in order to allow detailed performance analysis of the system under study (Yoo et al., 2012). Those metrics are also referred to as performance counters or hardware events. Performance counters, in this work, refer to relevant performance metrics on any layer of the system architecture. In ITIL, the measurement-based strategy is termed simulation modeling (Hunnebeck et al., 2011).

- **Model-based performance evaluation** refers to the creation of numerical or analytical models that allow to simulate or to calculate the behavior of a real system. This way, performance metrics can be projected under different conditions such as varying load factors or server capacity levels. Hence, this strategy principally does not require a running system to be set up. However, the crucial task here is to calibrate the model which may still involve measured data.

To conclude, well-designed prototypes may provide suitable performance assurance as part of measurement-based approaches but their setup is expensive and time-consuming (Chen et al., 2005). Particularly for EAs, tests must run across a variety of interdependent configuration options until a system is satisfactorily tuned (Chen et al., 2002). As a result, model-based techniques evolved to assess future system performance and traditional prediction literature heavily relies on queuing networks and discrete event simulation (Bichler et al., 2006). As pointed out by Becker et al. (2006), a combination of both strategies is conceivable too. For example, measurement data can be used to to validate performance models as done by Liu et al. (2004).

2.3.2 Classification of techniques

Either of the strategies outlined in Section 2.3.1 takes time and effort to prepare and execute. Hence, a number of aspects need to be considered when selecting an appropriate strategy for a given scenario. Those aspects include costs of modeling, ease of use, accuracy, flexibility, scalability, analyzability, and applicability (Becker et al., 2006).

Many concepts and methods related to model-based performance evaluation (cf. Section 2.3.1) are largely unknown to most IT practitioners (Menascé and Ngo, 2009) and, thus, are rarely applied in industry (Becker et al., 2009). Main reasons include a generally high time effort in construction and analysis of analytical models (Brosig et al., 2014) as well as essential expert knowledge about the model itself, the system to be implemented and its dependencies which may not be available (Westermann et al., 2010). Such modeling approaches obtain results through simulation or, if models can be formulated in a closed form, analytically. Due to the high amount of required domain expertise, respective techniques may be referred to as white-box and grey-box approaches. Although they can principally be applied already in the design stage of the software development life-cycle (SDLC), the credibility of derived insights sometimes remains questionable until their validation in later life-cycle phases. A prominent example for white-box modeling is the Palladio component model (PCM), proposed by Becker et al. (2009), which allows software engineers to model and analyze a component-based software architecture with respect to performance aspects before full implementation. Since PCM instances describe a full system architecture, they contain different models to cover aspects of the software implementation, its assembly context (system architecture), the deployment context (execution environment), and the usage profile (Becker et al., 2009; Krogmann et al., 2010). After these models were constructed and combined, the PCM instance can be transformed, e.g., to a an analytical solver or to a simulation model in order to make performance pre-

dictions. The method was also used by Brunnert and Krcmar (2015) for predicting the performance of EAs. Another example for white-box modeling, addressing the domain of SAP EAs, was proposed by Wilhelm (2003), who uses queueing networks to make performance predictions. The method relies on measurement data which is generated as part of benchmarks within running applications. A domain expert, who is familiar with the proposed approach, however, stated that modeling planned systems along with subsequent calibration of the model is simply too expensive to design a commercial offering on the basis of suchlike methods.[1] An example for a gray-box approach to predict average response times of transactions was published by Chen et al. (2002) and further elaborated in (Chen et al., 2005). Similarly to the work at hand, the authors particularly target EAs that utilize COTS. They build a non-linear mathematical model, whose coefficients are obtained during model calibration using real measurement data from a prototype. The approach requires the prototype to be as similar as possible to the system under analysis when applying the model. However, the authors state that the approach may be impractical unless application-specific characteristics will be incorporated into the model which are currently decoupled from the modeled infrastructure components. Furthermore, model validation was carried out using actual training data which diverges from the usual procedure. The idea to automate the construction of performance models for reusable components of an EA was also pursued by Kappler et al. (2008). The authors propose a method to build a PCM instance from JAVA code on the basis of a static code analysis. However, the determination of resource demands and execution times is still subject to subsequent dynamic analysis which must be carried out manually. Furthermore, the application's source code may not always be available. To overcome these drawbacks, Krogmann et al. (2010) propose a two-folded approach which was priorly described in Kuperberg et al. (2008): First, application bytecode is instrumented and, under different usage contexts, executed on a reference platform in order to count the number of instructions. The monitored data is used to feed a genetic algorithm which estimates bytecode counts on the basis of different input data. Based on the GA estimations, a platform-independent behavioral model is constructed. If combined with a platform-dependent performance model which is to be constructed in a second step, the models can be used to predict execution times. The approach effectively reduces the required domain expertise in the performance model construction phase. However, it covers platform-specific software behavior only to a limited extent as the two model types are created separately and actual execution times are not measured with the application under study. Furthermore, the approach requires access to the application bytecode and the integration of both models is still subject to manual effort. Hence, the approach may be classified as gray-box modeling. Another example of gray-box modeling was proposed by Cherkasova et al. (2009) who use a regression-based analytical approach to model CPU demands of application transactions on a given server configuration. The approach supports in detecting performance anomalies and CPU consumption changes but does not allow to predict actual response times.

[1] Expert interview with an SAP performance engineer at June 15, 2018

Measurement-based approaches, on the other hand, are more frequently used in practice than model-based approaches due to their effectiveness (Westermann et al., 2010) and straightforwardness. This rather practical strategy requires (parts) of the planned system (e.g., a prototype) to be implemented and measured. Measurements can be carried out during regular operations or as part of stress tests or performance unit tests inside quality assurance systems (Horkỳ et al., 2015). As introduced in Section 2.1.4, different means exist to monitor application performance. Obtained results can be analyzed easily and their credibility is unquestioned. Multiple iterations allow for testing various levels of workload and all executional dependencies are considered which model-based approaches cannot guarantee (Venkataraman et al., 2016). On the other hand, correction costs, in case of insufficient performance, are high and degrees of freedom with respect to alternative configurations are low (Brosig et al., 2014). As a costly consequence, performance requirements are often considered lately and performance testing is done when an implemented or relocated system is about to go live (Balsamo et al., 2004; Menascé, 2004). This practice was originally termed by Smith (1981) as *"fix it later approach"* but can be observed in today's projects either (Tudenhöfner, 2011). The correction of performance failures at that last stage is inefficient, expensive, delaying and professionally irresponsible (Menascé, 2004; Tertilt and Krcmar, 2011).

In contrast to white-box and gray-box modeling, black-box approaches significantly reduce the amount of domain expertise. Instead, models can be constructed by means of machine learning techniques. During the training phase, existing patterns and interrelations are learned from measurement data and fed into a performance model. The applicability of the model is limited by the variety of training data. Therefore, large amounts of performance counters are beneficial in order to obtain a suitable model. The data basis may be generated by running differently configured benchmarks against a prototype of the planned system. Hence, (parts of) real systems are required prior to model creation, adding implementation costs. Once the model was trained, it can be applied to project performance under varying conditions by passing different input features. Although models are supposed to generalize to unseen data (Mohri et al., 2012), configurations (e.g., a specific type of server) which heavily diverge from what was found in the training data cannot be used since their effect on resulting performance is unknown. Another downside is given by the limited analyzability of obtained results due to the black-box nature of the technique. This downside exponentially intensifies with a growing number of input features. For this reason, model validation is an important step in order to assess the credibility of gained insights. As an example for machine learning-based techniques, Duan et al. (2009) use a hybrid Bayesian-neural network in order to improve the effectiveness of scheduling systems by dynamically model and predict execution times. A similar approach was taken by Huang et al. (2010) who predict execution times using a non-linear regression model to be used by job schedulers. In 2011, Niehorster et al. use support vector machines (SVM) to predict the resource requirements of an application. They design a multi-agent system where every agent handles user requests and shares measurement data of the re-

quest in a global knowledge base for subsequent modeling. Gupta et al. (2008) predict time ranges for query executions in order to manage database workloads. In contrast, the authors of (Yoo et al., 2012) demonstrate that machine learning-based techniques can also be utilized to detect performance anomalies. Their system feeds Random forests with hardware counters in order to identify bottlenecks without providing architectural knowledge. As another example for performance projection, Venkataraman et al. (2016, p. 365) design a "performance counter based approach" in order to predict run-times of analytics jobs. The authors state that such approaches typically utilize advanced machine learning techniques such as Random forests and SVM to train prediction models. Finally, Matsunaga and Fortes (2010) test several machine learning techniques for their suitability to predict application performance and their resource usage. One key insight of the experiments performed by the authors is that the accuracy of machine learning techniques can be improved by including application-specific parameters besides plain hardware counters.

To summarize, the applicability of machine learning-based techniques is strongly limited by the variety of training data. Furthermore, time and costs associated with the collection of historical data represent major drawbacks. According to Venkataraman et al. (2016), the main challenge of this approach is to minimize the time spent on collecting and preparing training data in order to achieve a satisfying accuracy. Both of the mentioned limitations can be mitigated if measurement data was shared within a domain, e.g, the domain of customers of a specific COTS EA product. This way, models could benefit from increased variety of training data spanning a potentially large amount of design alternatives. Consequently, the search space that can be exploited by what-if questions is extended to scenarios that were not implemented by the respective performance engineer. Performance projection techniques which leverage the outlined potential, in this work, are classified as *machine-leaning based techniques using shared performance counters* (cf. Table 2.2 on the right). The most related approach was proposed by Ganapathi et al. (2009) who predict execution times of database queries in order to support workload management. They conclude with a long-term-vision of domain-specific performance models in order to support answering a variety of what-if-scenarios in the context of workload management, capacity planning and capacity management. This research gap is addressed by the thesis at hand for the domain of EAs. While the approach outlined by Ganapathi et al. is limited to the layer of database queries, a holistic projection of business transaction performance, covering all layers of the EA, was proposed in (Müller et al., 2017b) on the basis of experimental results published in (Müller et al., 2017c). Both publications represent preliminary works to this thesis. Early experiment results were also published in a master thesis, which was supervised by the author of this thesis, representing promising groundwork (Wirth, 2015). While the outlined technique may be applied in various use cases that require performance projections, in the context of this work, it is used to evaluate solution candidates of the SPP with respect to performance-related SLAs. Table 2.2 summarizes the preceding classification and discussion on performance projection techniques with respect to the outlined dimensions.

Technique	**Measurement-based**	**Model-based using simulation engines**	**Model-based using analytical solvers**	**Machine learning-based using isolated performance counters**	**Machine learning-based using shared performance counters**
Model type examples	None	Queueing networks, petri nets, markov chains	Closed-form expressions such as (non)linear models	Random forests, support vector machines	Random forests, support vector machines
Real system required	Yes	No	No	Yes	No
Domain expertise	High	High (white-box modeling)	Medium (gray-box modeling)	Low (black-box modeling)	Low (blackbox modeling)
Earliest phase in SDLC	Implementation	Design	Design	Implementation	Design
Required training data	No model training	None or low amount to calibrate models	Few user-generated or measured top level parameters	Large amount of low level counters	Massive amount of low level counters
Degrees of freedom	Low	High	Medium	Medium	High
Number of testable design alternatives	Limited to implemented designs	Limited to modeled designs	Limited to modeled designs	Limited to implemented designs	Limited to shared designs
Analyzability	High	High	Medium	Low	Low
Literature examples	(Menascé and Almeida, 2002; Menascé, 2004; Lilja, 2005; SAP, 2019a)	(Wilhelm, 2003; Bertolino and Mirandola, 2004; Bondarev et al., 2005; Becker et al., 2009)	(Chaudhuri et al., 2004; Chen et al., 2005; Cherkasova et al., 2009)	(Gupta et al., 2008; Duan et al., 2009; Huang et al., 2010; Niehorster et al., 2011; Yoo et al., 2012; Venkataraman et al., 2016)	(Ganapathi et al., 2009; Müller et al., 2017b,c)

Table 2.2: Classification of techniques for EA performance projection.

2.3.3 Supervised learning techniques

Supervised learning techniques, according to the classification provided in Section 2.3.2, belong to the class of machine learning-based black-box approaches. In the case of performance prediction, usually the mean response time or the throughput may be predicted. This information indicates the application of supervised learning techniques as opposed to unsupervised learning where the attribute of interest is not given.

In supervised learning, modeling refers to the creation of a function which maps a set of input features to an output value. Due to the high complexity of real-world phenomena, this function in many scenarios cannot be created manually. Instead, machines are used to learn patterns from the input data. Depending on the type of output, two classes can be distinguished: Classification techniques assume the output to be a discrete label such as fast or slow. On the other hand, regression techniques deal with a continuous output such as any real value between 100 or 500 milliseconds (ms) (Mohri et al., 2012). In the course of this work, response times are to be predicted. Therefore, few relevant machine learning techniques are introduced in the following, with a focus on regression.

Linear regression

A very simple example is linear regression. It provides a good basis of knowledge in order to understand more sophisticated techniques and therefore serves as a starting point on this occasion. In addition, many real world phenomena can be approximated using a linear model and even if they are not applicable for the global model, they may still be used as a sub-component of larger systems (Shalizi, 2006; Freitas, 2015).

In order to learn, input data is required during the *training phase*. The input data is also referred to as training data and comprises a data set of n observations x, here expressed as $x_{1:n} = [x_1, x_2, ..., x_n]$. Each observation consists of a number of attributes d, also called dimensions, which form the input. An additional dependent variable y builds the output that is to be described by the model. The input is also referred to as predictors, covariates or independent variables. The output is also referred to as target or dependent variable. After the model was trained, it is supposed to compute y for a new input x_{n+1} which was unknown during the training phase. In other words, the model must generalize to unseen data (Mohri et al., 2012) and the predicted output $\widehat{y}$ is a function $\widehat{y}(x_{n+1})$ of the new input. To achieve this, the influence of each input feature on the dependent variable is to be computed. This influence is expressed by the parameters β_1 to β_n and the resulting model, in its simplest form, would predict the output $\widehat{y}$ as follows:

$$\widehat{y}(x) = \beta_0 + \beta_1 x \tag{2.1}$$

The form of equation 2.1 indicates a simple line where β_0 defines the intercept with the y-axis and β_1 describes the slope of the line. An example is given in Figure 2.8. Here, the red data points represent 4 observations x_{1-4}, each having one input feature and the respective output y_{1-4}. The only parameter of a linear model that is not associated to

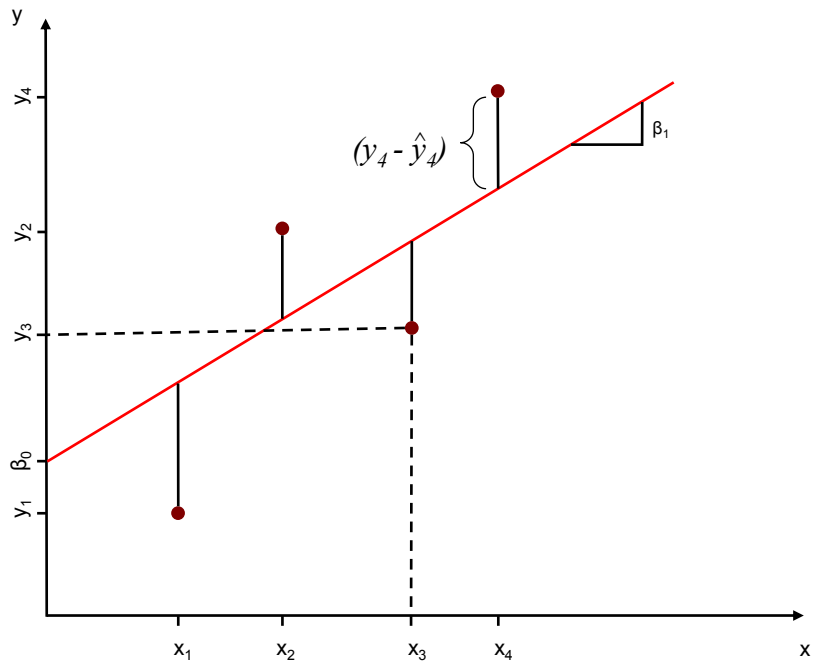

Figure 2.8: Linear regression sample using four observations, based on Freitas (2015).

any of the input features, β_0, represents a baseline that cannot be explained by a single feature. As an example, if the power consumption of a server is to be predicted, β_0 could be a portion of, e.g., 100 Watt that is always consumed in addition to the load-dependent portion. In Figure 2.8, the model fits the data points (purple red bullets) using the red line. In order to determine how well the model performs, an objective function is required. In linear regression, it is the goal to find a line which has minimized distance to the data points. This distance (also referred to as *error*) is highlighted in Figure 2.8 by the black lines and can be expressed by the depicted formula. Simply put, the error between the output in the training set and the predicted output should be as low as possible. In order to tolerate small distances (<1) and to penalize large distances (>1), it is a good practice to square the error. Hence, across n data points, the objective function is to minimize the sum of the squared errors as depicted in Equation 2.2 which utilizes $\widehat{y}$ from Equation 2.1:

$$J(\beta) = \sum_{i=1}^{n}(y_i - \widehat{y_i})^2 = \sum_{i=1}^{n}(y_i - \beta_0 - x_i\beta_1)^2 \tag{2.2}$$

The illustrated example uses only one feature per observation. If multiple features d_1 to d_n are to be considered, the modeling process is termed multiple linear regression; the resulting model is depicted in Equation 2.3.

$$\widehat{y}(x_i) = \beta_0 + \beta_1 d_{1i} + \beta_2 d_{2i} + ... + \beta_n d_{ni} \tag{2.3}$$

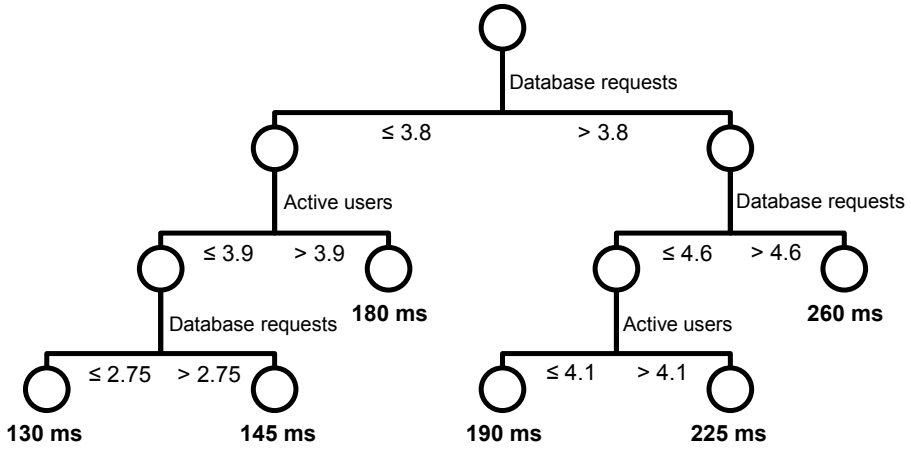

Figure 2.9: Regression tree sample with two input features, based on Shalizi (2006).

Regression tree

Scenarios which show a large number of input features that have complex non-linear relations with the dependent variable, can hardly be handled by simple linear models. A suitable way to handle the high-dimensional input space is to partition the data into sub-partitions. If this is performed iteratively, one can refer to *recursive partitioning*. The idea behind *regression trees* is that, while it is not possible to fit a global linear model, the same may perform well on a limited subset of the input data (Shalizi, 2006). Therefore, the data is split iteratively depending on individual criteria, resulting in a tree structure where the mentioned sub-partitions are represented by the terminal nodes, also referred to as leafs of the tree. A sample tree is depicted in Figure 2.9 with two input features describing the number of database requests and the number of active users in a system. The output, in this example, is a continuous value of the system's response time, measured in milliseconds (ms). The training data in this example may include the input matrix x and the output vector y from Equation 2.4:

$$X = \begin{bmatrix} 3.0 & 4.1 & 5.2 & 3.5 & 4.2 \\ 3.2 & 3.6 & 5.5 & 4.0 & 6.1 \end{bmatrix} \; y = [130, 170, 280, 210, 235] \tag{2.4}$$

The tree is grown until an additional split does not provide a sufficient amount of additional information. The so called *information gain* may be defined in different ways and is to be maximized when deciding how to perform the next split. Therefore, each split tries to separate the remaining observations as much as possible with respect to the target variable. One way, used in the original version of regression trees, is to minimize the sum S of squared errors of the target variable from their respective average value in the two resulting nodes. The feature whose split minimizes S is chosen to separate the data into two additional nodes, according to individual conditions as indicated in Figure 2.9 by the branch annotations. These conditions represent the borders of a partition; they are illustrated in Figure 2.10 by the gray lines that split the input space into 6 sub-partitions

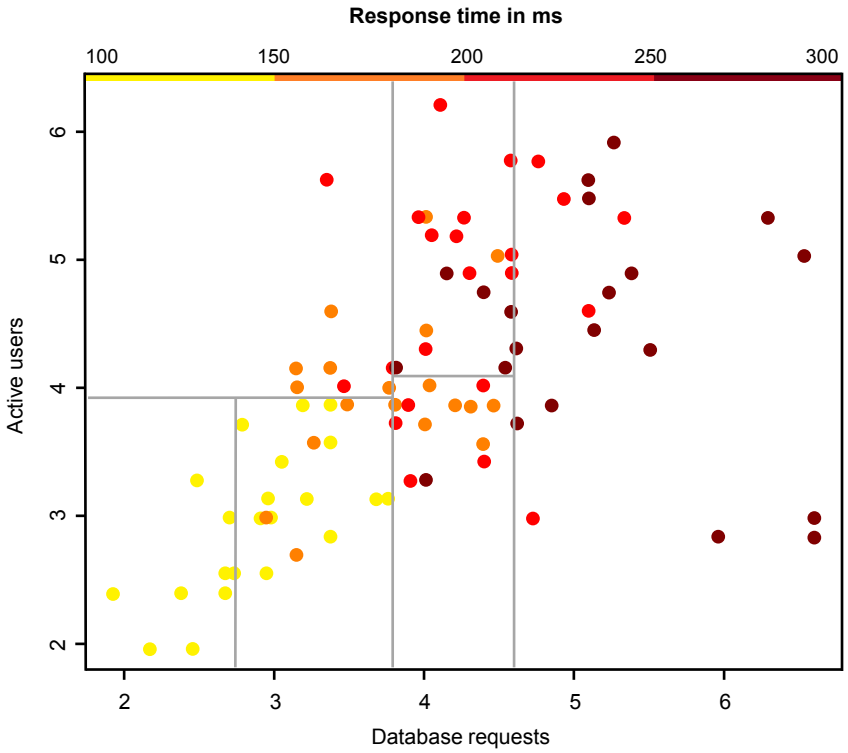

Figure 2.10: Recursive partitioning of a regression tree, based on Shalizi (2006).

which, in turn, represent the leafs of the tree in Figure 2.9. The input features of matrix X in Equation 2.4 are shown in Figure 2.10 on the x- and y-axis. A third dimension, the output variable y, is indicated by the color scheme. The total area represents the data in the root node. As the tree grows, the differently colored data points are to be separated. A local splitting decision, once it was taken, is never questioned even though it may not have been optimal from a global point of view. Due to this behavior, regression trees belong to the class of greedy algorithms. According to Shalizi (2006), the fulfillment of one of the following criteria stops the tree from growing:

- All data points in a node show the same value for all input features
- The sum of squared errors cannot be further decreased beyond a defined threshold
- The number of data points in one of the resulting nodes falls below a defined threshold

Additionally, some implementations include a parameter which controls the maximum depth of a tree. If a node is not split any further, it becomes a leaf. Finally, on each leaf, a regression technique such as a linear model is applied in order to perform the prediction. In the original version of regression trees, a constant estimate of the target variable is made by averaging across the observations within the leaf. It is to be noted that a model is only valid inside the associated partition.

Random Forests

Models that show high variance in terms of the predicted output may have been *overfitted* to the training data. This risk is also entailed by regression trees, especially if trained with a high depth. Therefore, regression trees do not tend to generalize well to unseen data. A common strategy to decrease the variance and therefore to avoid overfitting is *bagging*. Here, multiple learners of the same type are trained on randomly selected subsets of the input data and the prediction result is simply the averaged result of all learners. The added randomness reduces the variance and creates a more stable model. This strategy was leveraged by Breiman who proposed the concept of Random forests (RF) in 2001. RF combine a defined number of regression or classification trees which were trained using a subset of input features, randomly selected (with replacement) for each split decision. This is sometimes called *feature bagging* and reduces the correlation of the trees. The type of model belongs to the class of *ensemble learning techniques*. RF are known to be one of the most powerful classifiers; they are used in many real-world applications such as, e.g., in Microsoft Kinect to determine the position of the hand (Freitas, 2013). In general, a Random forest where each of the trees uses a random subset of features shows higher accuracy than a single tree using the full set of features (Criminisi et al., 2012). An advantage of the bagging strategy lies in the ability to parallelize the training phase of the individual learners since their results do not depend on each other. Just like regression trees, RF can deal with complex nonlinear relations by splitting the problem into smaller sub-problems (cf. Figure 2.10) that can be solved more simply (Criminisi et al., 2012). Therefore, the general procedure is to select a subset of features, perform a split that maximizes information gain and, in each of the created partitions, select a new feature subset to proceed with. This is performed for each tree until one of the stop criteria, listed earlier in this section, takes effect. On the terminal nodes, different predictors can be used such as constant or linear models (Criminisi et al., 2012). Finally, the prediction result y is computed as the average of all trees t (cf. Equation 2.5).

$$y = \frac{1}{T} \sum_{t=1}^{T} y_t \tag{2.5}$$

Important model parameters include the outlined stop criteria as well as the number of randomly selected features which are tested in order to decide for a split. In the case of regression with p features, $\frac{p}{3}$ represents a size of subset which usually performs well (Liaw and Wiener, 2018).

Support vector machines

The idea of partitioning the input data is forged ahead with the concept of Support vector machines. The theoretical foundation was originally built by Cortes and Vapnik under the term Support-vector networks and demonstrated for pattern recognition (Cortes and Vapnik, 1995). However, two years later, Drucker et al. (1997) clarified how the

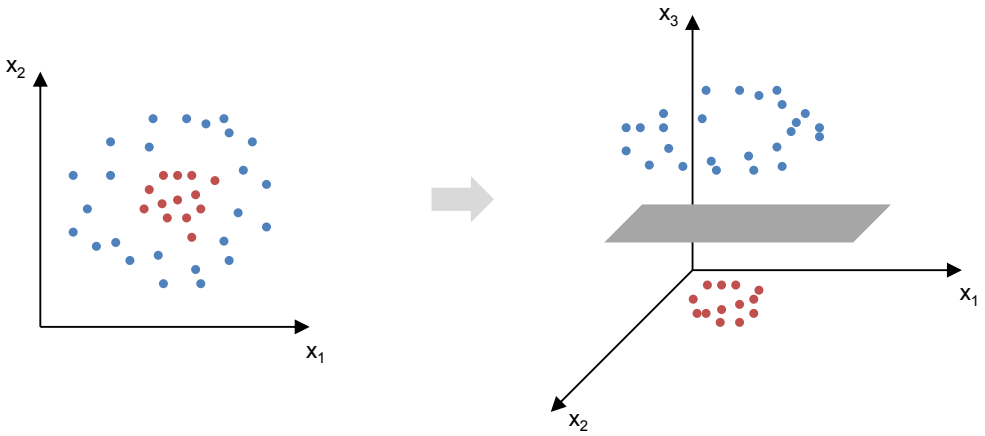

Figure 2.11: Kernel trick using a polynomial function in SVM, based on Jordan (2004).

concept can be utilized for regression. The basic concept relies on the idea to transform input features into a high-dimensional feature space in a way that data points become separable. Depending on the number of dimensions, splitting entities can be imagined as a simple line (two dimensions), a plane (three dimensions), or a hyperplane (more than three dimensions). As opposed to Regression trees and Random forests, splits are performed by maximizing the margin between the hyperplane and closest points which are termed support vectors. This can be formulated as a constrained optimization problem and solved by means of Lagrange multipliers (Burges, 1998). However, the mathematical formulation is out of scope in this work; it can be studied in more detail in (Rockafellar, 1993) and, in the context of SVM, in (Suykens and Vandewalle, 1999).

The transformation of input features is performed by a kernel function which can take different forms. A simple example is to add an additional dimension by squaring the input dimensions so that data points on the new axis become positive. The respective kernel function is formulated in Equation 2.6.

$$y(x_1, x_2) = y(x_1, x_2, x_1^2 + x_2^2) \tag{2.6}$$

While the example uses a polynomial kernel (as also originally applied in (Cortes and Vapnik, 1995)), alternative options are, e.g., a linear kernel, a radial basis function, or a sigmoid kernel. The transformation of the input space is referred to as *kernel trick* as it allows to separate data points by increasing the dimensionality. A visualization of Equation 2.6 is given in Figure 2.11, where a function which considers two input features (x_1 and x_2) is transformed into a function of three features. As opposed to the original representation of data points (Figure 2.11 on the left), the transformed feature space allows to build a plane that separates the data points (Figure 2.11 on the right).

Depending on the data, strict separations as illustrated in Figure 2.11 may not be possible. In this case, misclassifications must be allowed up to a defined threshold. The threshold can be controlled over a parameter C which defaults to 1. A higher value

fosters hard margins which may result in overfitting. In contrast, low values lead to soft margins that allow misclassifications in favor of improved generalizability. Therefore, the optimization of hyperparameters, when applying SVM, includes the parameter C and the selection of a suitable kernel version.

Boosted trees

Boosting is another ensemble technique which combines a number of learners as also done by bagging techniques such as RF. Boosting, however, relies on a rather evolutionary approach where the overall learning process is guided on the basis of preceding results. This way, subsequent learners are forced to focus on records of the training data on which previous learners struggled to achieve low errors. According to Schapire (2013), the first practical boosting algorithm is AdaBoost. In its original algorithm formulation, Freund and Schapire (1997) use a function *WeakLearn* that refers to any learner which can be used in conjunction with AdaBoost. As the name suggests, weak learners are expected to be only slightly better than random guessing. An example of a weak learner is a regression tree with a limited depth which is therefore also called *stump*. They are added to the learning process successively and trained on the data which was weighted according to previous errors. While, initially, each record is equally weighted, the weightings are scaled after each learning iteration using an exponential error function (cf. Equation 2.8).

The process terminates after T learning iterations. In contrast to Bagging, when used for regression, the overall prediction output is a weighted median of all weak learners, depending on their total error (Drucker, 1997). The weighting is carried out using a logarithmic function which grants learners with low errors a high influence on the result while weakest learners receive negative influence. Accordingly, the weight α of a learner in iteration t is calculated as depicted in Equation 2.7.

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - e_t}{e_t} \right) \tag{2.7}$$

These weights are computed within each iteration and are then used to guide the subsequent learners. Each record i of the training set is weighted according to a function which Equation 2.8 depicts in a simplified way for classification problems.

$$w_{t+1}(i) = w_t(i) \cdot e^{\alpha_t \cdot e_t} \tag{2.8}$$

In this example, the error e within an iteration t is 0 if the prediction was correct. In all other cases, it accounts to 1. Hence, if a good learner (high α_t) causes high errors on particular records, increased weights will cause subsequent learners to focus on those records. If, on the other hand, the error was 0, weights will remain with no change (Brownlee, 2016). Depending on the specific AdaBoost implementation, variations of Equation 2.8 are used. Similar to RF and different from SVM, when using AdaBoost in conjunction with decision trees or regression trees of a limited depth, most relevant features are to be selected to split the data set as described in Section 2.3.3. Hence, AdaBoost

deals well with the curse of dimensionality and is known to work well without much tuning when compared to other ML algorithms (Kégl, 2013; Caruana and Niculescu-Mizil, 2006). Since Boosting is designed to explore shares of the training data which are particularly hard to explain, AdaBoost is sensitive to outliers which are to be excluded beforehand. As opposed to RF, the learners cannot be trained in parallel (Friedman et al., 2017, p. 343 et seqq.).

2.3.4 Model validation

According to ITIL, the accuracy of a model depends "[...] on the skill of the person constructing the model and on the information used to create it" (Hunnebeck et al., 2011, p. 92). Good quality with respect to both aspects can be ensured by following processes in the domain of data mining. Widely established examples are the Cross-industry standard process for data mining (CRISP-DM) and the Analytics solutions unified method for data mining (ASUM-DM). The steps of CRISP-DM are depicted in Figure 2.12. ASUM-DM, in contrast, incorporates additional steps, required to build solutions that incorporate models as well as their operation and maintenance in an agile manner (IBM, 2016). Since any data mining process must fulfill a given purpose, it is the first step to define business objectives along with data mining goals. Subsequently, data sources are to be identified and initial data must be collected, described and explored. Additionally, the data quality is to be assessed. The third step, data preparation, incorporates techniques to select, clean and format relevant data (Chapman et al., 2000). The data builds the input space which comprises features to feed models with, as described in Section 2.3.3.

An essential part of any data mining process is the model validation (cf. step five in Figure 2.12) (Shaeffer, 1980). In here, it must be tested how well the model performs when applied to new scenarios. Since models are supposed to generalize to unseen data, validation must be carried out using a data set different from the training set. This data set is referred to as *test data* or *validation data* and ensures that a given model does not overfit to the training set. In addition, so called *cross-validation* implies that multiple validations are carried out using different subsets of training and test data. Typically an average value across the resulting errors represents the overall accuracy of the validated model. While cross-validation in most cases is beneficial to obtain an unbiased error from the test set, some model types fulfill this task internally during the training phase. As an

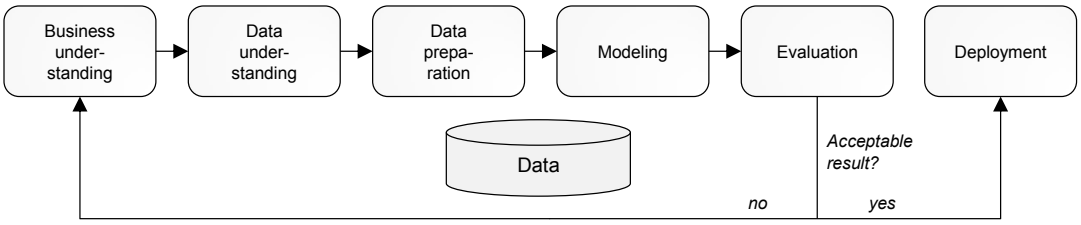

Figure 2.12: The CRISP-DM methodology, based on Chapman et al. (2000).

example, in Random forests, each tree is already built using a different bootstrap sample of input data where usually around one third of the data is not used (Breiman and Cutler, 2019). To measure model accuracy, various metrics exist. The simplest error metric is the mean error (ME) which represents the mean difference between predicted values $\widehat{y}$ and measured values y across the test set n_{test} as given in Equation 2.9. Similarly, the mean absolute error (MAE) refers to the modulus of the error as given in Equation 2.10.

$$\text{ME} = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} (\widehat{y}_i - y_i) \tag{2.9}$$

$$\text{MAE} = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} |\widehat{y}_i - y_i| \tag{2.10}$$

To indicate model accuracy, often a percentage value is used since the same are easy to interpret and comparable across models of different magnitudes (Mayer and Butler, 1993). Accordingly, the mean absolute percent error (MAPE) expresses the mean distance between measured and predicted values in relation to the measured values (cf. Equation 2.11). Particularly in business context, the MAPE is a frequently used error metric (Gneiting, 2011).

$$\text{MAPE} = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} \left| \frac{\widehat{y}_i - y_i}{y_i} \right| \tag{2.11}$$

Another metric, more frequently used in the domain of statistics, is the root mean squared error (RMSE) which represents the root of the mean squared errors (cf. Equation 2.12). As pointed out in Section 2.3.3, squaring of errors is beneficial in order to penalize large errors while tolerating low errors and, here, causes the RMSE to be always positive. The lower the value the better and an RMSE which amounts to 0 indicates a model with highest possible accuracy. The RMSE is comparable only across data of the same magnitude as it is based on the ME which scales depending on the data.

$$\text{RMSE} = \sqrt{\frac{1}{n_{test}} \sum_{i=1}^{n_{test}} (\widehat{y}_i - y_i)^2} \tag{2.12}$$

The RMSE of a model may be compared to the standard deviation of the data in order to evaluate whether a model performed better than a simple guess on the basis of the average value across all observations. A similar purpose is pursued by the coefficient of determination which is also termed R^2. Here, variation is put into ratio with the sum of the squared errors. According to Equation 2.13, R^2 approaches 1 if the distance between predicted values and measured value is lower than the distance between the measured values and their average value. Hence, models with higher R^2 (closer to 1) are to be preferred.

$$R^2 = 1 - \frac{\sum_{i=1}^{n_{test}} (y_i - \widehat{y}_i)^2}{\sum_{i=1}^{n_{test}} (y_i - \bar{y}_i)^2} \tag{2.13}$$

In general, thresholds of acceptable values must be defined by the modeler in accordance

to requirements inherent to the application area of the model (Almeida, 2002). If results are not satisfying, tuning activities require to iterate the process once again (cf. Figure 2.12). As an example, input data may be filtered or enriched or a different set of features may be selected. Depending on characteristics of the data, hyperparameters of the model could be optimized or the type of model may have to be changed. If, on the other hand, acceptable errors are computed, trained models are ready for deployment.

2.3.5 Summary

The performance of an EA is to be projected before entering the operations phase in order to avoid SLA penalties and expensive correction costs. As part of the capacity management, various what-if scenarios form the input to model-based, measurement-based or machine learning-based techniques. If applicable, machine learning-based techniques that utilize shared performance counters combine the main advantages of model-based techniques (applicable in the design phase with high degrees of freedom) while avoiding main disadvantages of measurement-based techniques, since the amount of testable design alternatives is not limited to implemented but to shared designs. Here, techniques from the field of supervised machine learning are employed. Random forests, Support vector machines, and Boosted trees represent commonly used learners that deal well with nonlinear interrelations in a high-dimensional feature space. Required activities to prepare the data and train models are specified, e.g., in the Cross-industry standard process for data mining. This process incorporates the model validation as a crucial task to assess the credibility of obtained results. A widely used error metric that is comparable across varying data and model types is the mean absolute percent error which relates distances between predicted and measured values to measured values. Acceptable error ranges are strongly domain-specific and models may be tuned in multiple iterations until their deployment.

2.4 Related research artifacts

Existing server consolidation and service placement approaches represent excellent examples of interdisciplinary research as various mathematical or algorithmic solution techniques are applied to optimization problems in order to address cost saving objectives in the domain of business informatics. This section analyzes and classifies related approaches which address the research goal of this thesis.

2.4.1 Overview of related approaches

With the increasing spread of cloud computing, the problem of placing virtual entities on physical servers became a widely studied field. While, initially, server consolidation used to be a manual task, first approaches evolved to automate the steps of placing services on available servers in 2001. A comprehensive basis of related work was investigated by López-Pires and Báran in a literature review on virtual machine placement. In total, the authors identified 446 research articles using Google scholar; those were subsequently filtered and

grouped by the publishers IEEE, Springer, Elsevier and ACM. The remaining 172 articles were further filtered through abstract reading with respect to a clear VMP focus. In addition, short papers with less than six pages were excluded, resulting in a remainder of 84 relevant articles. Finally, López-Pires and Báran classify the publications with respect to the optimization approach, the objective function, the solution technique and the problem formulation. The latter determines whether the problem is solved online or offline. Other classifications refer to this aspect as the time of decision making (Varasteh and Goudarzi, 2017) which highly depends on the technical architecture of the solution environment along with the employed virtualization technique (cf. Section 2.2.2). Regarding the objective function, consolidation approaches most commonly address one or more of the following objectives (López-Pires and Báran, 2015):

- Group 1: minimization of energy consumption (50%)
- Group 2: maximization of network traffic (30.9%)
- Group 3: maximization of economical revenue (22.6%)
- Group 4: maximization of performance (16.7%)
- Group 5: maximization of resource utilization (15.5%)

According to this classification, the thesis at hand addresses the two least studied groups, since the artifact is designed to maximize resource utilization while maximizing resulting performance. Therefore, approaches that are formulated as offline problems and classified into objective function groups 4 and 5 are considered to be most related. As the qualitative review of López-Pires and Báran provides a valuable basis of related literature for the present work, all articles classified as offline problems were studied. Additionally, those articles were filtered with respect to the targeted objective functions; the result served as a baseline for forward-backward-search in order to sprawl the search space and to cover the chronological gap between 2015 and today. While herein the focus again was put on offline problem formulations, online approaches were additionally studied if they deal with impacts on the service performance. The forward-backward search revealed an additional literature review, conducted by Mohamadi Bahram Abadi et al. (2018), covering only recent articles that were published after 2012. As the authors also provide a classification regarding the problem formulation, their studied articles could be filtered for offline problems (here termed “static consolidation techniques”). The resulting set of publications was included into the search process of this work too. The overall process, as depicted in Figure 2.13, gained a sufficient overview of the state of the art in the domain of service placement. In the following, most related artifacts are briefly introduced in chronological order with a focus on strengths to be built upon and weaknesses to be obliterated.

In 2001, Pinheiro et al. propose a new research direction that considers a systematic load consolidation in order to reduce energy consumption within clusters. The publication represents one of the first works in this field as pointed out by Beloglazov et al. (2012). In the article, a trade-off between power and performance is introduced. The authors state

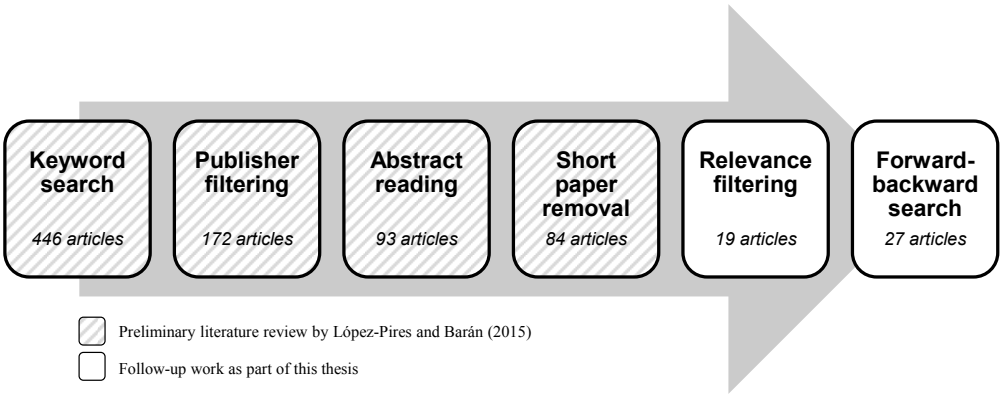

Figure 2.13: Literature review process, based on López-Pires and Báran (2015).

that power consumption decreases with the number of running nodes in a cluster while performance suffers when nodes are over-utilized. Two metrics for performance measurement are distinguished: throughput and execution time. The former is used by their solution algorithm which predicts expected performance degradations for consolidation decisions based on portions of historical resource demands of the workload that is to be moved. The level of acceptable performance degradation must be specified by the cluster administrator in advance and the solution approach is limited to homogeneous machines. The algorithm makes decisions to turn off cluster nodes (to save power) or to turn on additional nodes (to handle load) and handles subsequent load redistribution. While percentage-based performance degradations are predicted with respect to throughput and execution time, the prediction strategy relies on a rather rudimentary calculation. For example, if one cluster node demands for 40% of the total bandwidth and an additional node claims 80% at the same time, execution time degradation of 20% is predicted. In this regard, Pinheiro et al. assess their execution time predictions to be inaccurate.

In 2005, Rolia et al. propose a capacity management service for EA clusters. The considered daily workload patterns lead to a two-dimensional bin packing problem with one time dimension and one resource dimension. The authors use a GA to optimize the service allocation; it is evaluated by performing a case study with 26 servers where the number of CPUs (16, 32, 64, or 128) represent the capacity limit in relation to a static reference CPU speed. While heterogeneous CPUs were assumed, other resource demands, such as main memory, were considered in the problem definition but were not included in the evaluation. Instead, sufficient main memory capacity of the servers was principally assumed. With respect to the service performance in the target design, the authors distinguish two service classes to represent production and non-production workloads. Furthermore, they define that interactive and batch workloads can have differing resource access probabilities. The overall objective is to prevent degraded resource access from causing increased application response times. Therefore, during evaluation, a spare CPU capacity of 50% in relation to the targeted utilization was introduced for all interactive workloads. While the strategy of spare capacity may keep response times low, a rather large portion of saving potential is

left unaddressed. The approach is based on groundwork presented by Rolia et al. (2003). Here, too, a soft assurance for quality of service was proposed by introducing additional portions of spare capacity amounting to either 50% or 20% per CPU or server.

One year later, Bichler et al. (2006) formulate two related problems for server consolidation which particularly address the domain of COTS EA: The Static server allocation problem (SSAP) and the Static server allocation problem with variable workload (SSAPv). Average CPU utilization in a 5-minute interval, measured over one month, was aggregated to 24 hourly demands of a typical day for each service. The resulting sequences serve as workload input to the formulated problem. The applicability is limited to homogeneous data centers with servers having identical capacities. Depending on the problem size, the used algorithm takes 5 minutes to several hours to solve the problem. For evaluation, data of 30 servers from a data center provider was used. The authors mention one placement constraint of a service that must be placed on a particular server. However, the fulfillment of the constraint must be accomplished as part of a manual pre-processing task that involves decreasing the capacity of the chosen server's capacity. The performance of services in the resulting design was not discussed.

Similarly to the proposed approach by Rolia et al., Cherkasova and Rolia (2006) develop a GA to optimize existing allocations. Evaluation was performed using data from 26 applications. To apply the approach, the server's CPUs are required to be homogeneous but their number can vary for each server. In order to reflect service performance, product owners of an EA are requested to manually define utilization ranges, resulting in acceptable and degraded performance, for a defined percentage of measurements in the historical workload trace. This way, the amount of spare capacity can be adjusted for specific services and time intervals, which, however, is a critical manual task whose success depends on the experience of the actor. In 2009, Cherkasova et al. proposed a component that predicts CPU demands by means of a regression method and fed the result into a queueing model with the objective to predict the maximum achievable throughput of a system.

Hyser et al. (2007) develop the prototype of a system which controls the mapping of VMs to PMs. The system carries out VM live migrations in accordance to load balancing policies within a domain of homogeneous physical servers. In order to make placement decisions, an automatic controller considers resource demands of VMs in terms of CPU, memory, network input/output (I/O) bandwidth and disk I/O bandwidth. The optimization problem is formulated as an online problem and solved using a version of the Simulated annealing algorithm which belong to the class of metaheuristics. The authors evaluate their prototype with experiments on four identical PMs. Placement constraints are not mentioned in the experiments, but their importance for future work is emphasized in order to meet existing SLAs which result from higher level business requirements.

In the same year, Wood et al. (2007) state that VM migration decisions are (at that time) predominantly made manually. Therefore, the authors develop a method, termed Sandpiper, that monitors resource usage and detects so called workload hot spots. In this case, VM migrations are initiated according to a newly calculated mapping of VMs

and PMs. In this sense, hot spots occur whenever the utilization of monitored resource dimensions exceeds a threshold for a defined time period. They are detected by means of a queueing model which estimates peak demands with respect to CPU and network bandwidth. The authors distinguish black-box and gray-box monitoring techniques as also referred to in this thesis in Section 2.1.4.

Bobroff et al. (2007) apply a first-fit heuristic to a bin packing problem which describes the mapping of VMs to PMs. They define static server consolidation problems to utilize a single average value per resource on the basis of which offline migrations are carried out. In the thesis at hand, this definition is followed with respect to the workload. However, online migrations may also rely on static workloads. Dynamic workloads, in this sense, must be defined as a sequence of resource demands over time, in contrast to a single static value. While some publications agree with this, others use the term dynamic to describe a flexible target design that can be adopted online. The authors furthermore state that performance-related SLAs are typically expressed as response time guarantees. However, their approach translates response time SLAs for business processes into CPU guarantees of each VM so that the resource demand exceeds the capacity in, e.g., no more than 5% of measurement intervals. The approach involves a prediction model which forecasts the resource demands for CPU and memory on the basis of historical workloads. While the evaluation of the algorithm is carried out using workload traces from real applications, only a single resource (CPU) is considered in a homogeneous and rather small environment using three IBM blade servers as PMs and 5 virtual machines. The integration of additional resource dimensions is declared as future work. The authors mention different reasons for isolation of services such as security, resource contention, and a co-sensitivity to patches and versions.

In 2008, Kusic et al. propose a resource provisioning framework that performs sequential optimization using a look-ahead control scheme. The objective is to maximize profit that is generated by trading services. Relevant parameters are formed by the number of VMs for each application, the total number of physical servers, and the CPU share for each VM. Those parameters are dynamically tuned and placement decisions are based on the revenue generated by the trading service to be placed. For this purpose, a pricing scheme is maintained and achieved response times are mapped to a dollar value, which the client is willing to pay. The approach is carried out online in the sense that the formulated placement problem is solved periodically. Herein, costs for starting and stopping machines are considered and SLAs represent response times which are modeled as a function of workload intensity. This function is obtained via simulation-based learning from measured response times on different CPU capacities. The number of SLA violations depends on a risk-preference function of the controller. While violated SLAs lead to refunds, satisfied SLAs yield rewards. The authors test their approach in a two-tier EA environment comprising an application and a database tier with heterogeneous servers, using a stock trading software. On average, 22% energy savings were achieved in their test environment. While service placement strongly depends on the revenue of each service,

additional constraints were not considered.

In a three-tier-approach called AutoGlobe, Gmach et al. (2008) calculate resource allocations based on complementary load characteristics. In a first step, the authors use a Lloyd-Max quantizer to cluster the services into periodic and nonperiodic workloads. While periodic workloads can be placed offline, an additional online resource management controls the response time at run-time and reacts to exceptional situations such as unforeseen usage fluctuations or hardware failures. Appropriate actions are identified online by a fuzzy controller and include the addition of service instances or the movement of services to more powerful servers. A parameter that controls the allocation limit can be defined manually using an upper utilization bound, e.g., in percentage of processor load. If the parameter is exceeded, an overload situation is proclaimed. For the initial placement, low-level technical metrics such as CPU and memory demands are used; user-level SLAs are not considered. Instead, SLA control is enforced as part of the subsequent operations phase through request scheduling within the limits of the chosen design. The authors suggest to complement their rule based controller (fuzzy logic) with a performance model to predict the quality of service which would improve the decision quality with respect to various what-if-scenarios.

In 2009, Van et al. propose the concept for a resource manager that optimizes the degree of SLA fulfillment and operations costs. The components of the online approach are part of an autonomic control loop. A local decision module controls the mapping of applications to VMs and a global decision module handles mapping of VMs to PMs by communicating with the hypervisor. A performance model was obtained from experimental data; the model provides response times for given workload intensity and CPU capacity. However, the model is represented by a simple table that maps a discrete number of arrival rates for different CPU capacities to resulting response times. If those values are not matched by real-world continuous workload data, the response time is calculated as the average of the two nearest tuples. Other application-level metrics or HW characteristics of the server are not considered although, as stated by Pinheiro et al. (2001), execution time depends heavily on application characteristics. VMs are required to follow classes of predefined sizes in terms of CPU and memory capacity. The evaluation is carried out using only VM classes where CPU capacities equal memory capacities, effectively resulting in a problem with one resource dimension. The mapping of VMs to PMs is carried out by a packing module that utilizes a constraint programming approach. The authors argue that, in contrast, heuristics would not be suitable since placement criteria may change and entailing adaptions of the heuristics are to be avoided.

A multi-objective optimization problem which simultaneously minimizes the total resource wastage, the power consumption, and the thermal dissipation costs was formulated by Xu and Fortes (2010). The authors claim the objectives to be conflicting although minimization of resource wastage and power consumption are, to some extent, mutually supportive. A grouping GA is used to solve the problem and both CPU and main memory are considered as resource dimensions. However, the workload is static and in the con-

ducted experiments, CPU capacity in GHz equals memory capacity in GB for all cases. Neither service performance nor placement constraints were considered.

In the same year, Boloor et al. (2010a,b) deal with request scheduling in geographically distributed data centers. The authors introduce a response time constraint to ensure SLA conformance. Request allocation is handled by means of a lookup table and a rank-based scheme which favors servers that are pre-loaded with required context data. While the formulation of a percentile response time SLA represents valuable related work, the concept rather addresses research gaps in the domain of online load balancing and, therefore, may complement an offline server consolidation approach during the operations phase.

Stillwell et al. (2010) define a resource allocation problem with a static but multidimensional workload. The authors develop several approximate solution approaches such as greedy heuristics and a GA in order to solve the problem. Higher level user metrics such as the response time are mapped to resource fractions assigned to a service by means of a linking metric. The metric is called yield and quantifies for each service how much of the resource demand is actually satisfied. In the allocation problem, the minimum yield over all services is maximized with the idea to consider performance in a fair manner. However, the yield does not quantify actual response time estimates.

Another publication of the year 2010 propose a linear programming (LP)-relaxion-based heuristic to solve the aforementioned SSAP and SSAPv (Speitkamp and Bichler, 2010) which was formulated by Bichler et al. (2006). After an LP-relaxed solution is found in the first step, fractional assignments are combined greedily as a second step. The authors use real workload data from 259 ERP servers and 160 web servers to evaluate the approach. The components were monitored for a period of three months in intervals of 5 minutes; resulting traces were aggregated to daily workload profiles which contain 24 hours, each represented by the 0.95-quantile. Thus, the authors decided to exclude peak demands due to their rare occurrences. They argue that overload situations would result in increased response times but do not quantify this statement; for decision makers, it remains unclear to which extent response times are affected. Both workload demands of running services and server capacity limits were measured in SAPS (cf. Section 2.1.1). Additional resource dimensions such as the usage of main memory were disregarded. The computational complexity appears to be worse than exponential and depends heavily on the problem size. The approach is therefore limited to an upper bound of about 700 servers. However, as stated in Section 2.2.1, enterprise-class data centers are formed by hundreds to thousands of servers (Brown et al., 2007; Johnson and Marker, 2009; Jennings and Stadler, 2015) and continue to become the prevailing hosting facility (Carr, 2005; Ng et al., 2018). Possible placement constraints were listed but not considered in the experimental evaluation.

Feller et al. (2011) formulate a multidimensional bin packing problem to minimize energy consumption of data centers and develop a metaheuristic, based on ant colony optimization (ACO), to solve it. As part of the simulation-based experimental evaluation, multiple types of resource are considered and the used workload is dynamic, thus, includes

a sequence of time intervals. However, the approach requires all considered servers to be homogeneous. Service performance and placement constraints were not considered.

In 2012, Beloglazov et al. developed an online VM consolidation approach which takes SLA violations into account. The VM placement is carried out by means of a modified BFD algorithm (cf. Section 2.2.4). Herein, the approach considers solely the CPU as the main energy consumer. However, the authors emphasize the need to consider main memory as the next largest power consumer as an essential open challenge. SLAs are reflected by utilization thresholds and policies whereby the provider is requested to define allowed performance degradation levels in percent.

One year later, another multi-objective optimization problem was formulated by Adamuthe et al. (2013). The objectives are to maximize profit, to maximize load balance and to minimize recourse wastage. The problem is solved using variants of a genetic algorithm and considered resource dimensions are CPU, memory and network bandwidth. For their experiments, the authors generate data and tested the approach for up to 60 VMs that are to be placed on homogeneous PMs. The formulated placement constraints are limited to the ones which are inherent to the problem: A capacity constraint and a placement guarantee constraint define that every VM must be placed while servers must not be overloaded with respect to any resource.

In contrast, Sun et al. (2013) aim at minimizing the number of PMs while achieving an optimal resource utilization across the occupied PMs. The solution technique is based on a matrix transformation algorithm and the considered resource dimensions include CPU, main memory and disk. The authors show initial configurations of VMs and PMs before repeating experiments over an increasing number of VMs but it remains unclear which configurations were used in the experiments and if the PMs are allowed to be heterogeneous. The problem formulation comprises three constraints which, however, simply require the considered resources to be not overloaded.

Gao et al. (2013) minimize power consumption and resource wastage at the same time. Similarly to the work proposed by Feller et al. (2011), an ACO metaheuristic was applied to a test set comprising 200 VMs and 200 PMs. It uses a multidimensional resource vector containing static CPU and memory demands. Although the authors claim to support heterogeneous servers, the 200 PMs used in the experimental evaluation are homogeneous. The evaluation shows that the solution performance achieved by the algorithm is similar to a grouping GA. The performance of the placed services is not explicitly considered but a threshold value ensures that an upper bound of server capacity is not exceeded in order to avoid severe performance degradations.

Also in 2013, Hong et al. develop a VM placement approach that aims at maximizing the profit of cloud gaming providers while minimizing the response time which they call quality of experience (QoE). Accordingly, the utilized best-fit heuristic is termed quality-driven heuristic (QDH) and places a VM on the first suitable server of a list that is sorted by network latency in ascending order. While the approach scales with a large number of servers (solutions are calculated within less than 7.15 seconds for a system with more

than 3000 servers), it is limited to homogeneous machines.

Ankit et al. (2013) optimize virtual machine placement while supporting performance-related SLAs. The authors follow an online approach and quantify the overhead of live migrations using relative weights of different resource requirements. The resulting objective function requires to minimize the number of hosts for the final allocation and to minimize the migration overhead at the same time. This way, application performance is not affected significantly during relocations. Since only the number of hosts in the target design is pivotal, a homogeneous bin composition is assumed. The authors use an integer linear programming formulation for small problem instances and an FFD to solve cases with larger problem sizes. Placement constraints as listed in Section 2.2.5 were not considered. While the authors study the impact of the virtualization overhead on performance related SLAs, the effect entailed by the identified solutions during their operations phase was not measured and therefore considered only indirectly.

SLAs are also considered by Pires and Barán (2013) who propose a multi-objective VM placement approach. Therefor the authors define a service level agreement provision constraint with rather simple implications: services are classified to be critical or not by means of an additional parameter SLA which is either 1 or 0. Critical VMs must be placed on a host while non-critical VMs may not be placed on any host. A memetic algorithm is used to solve the problem. Although the formulation allows for servers with varying CPU, memory and storage capacities, only homogeneous servers were used in the experimental evaluation. While the multi-objective problem formulation is capable of considering three objective functions simultaneously, it is impractical to map SLAs which must be expressed in a clear and measurable way (cf. Section 2.1.1) to a single boolean parameter. Hence, the approach surely is mathematically meaningful but practically inappropriate.

Another VM placement solution, termed BtrPlace, was proposed by Hermenier et al. (2013). The approach relies on the concept of constraint programming which guarantees to find, if present, a globally optimal solution. The approach is tested in multiple cases, each time using homogeneous servers. The considered workload is static and implies CPU and memory demands. With respect to placement constraints, BtrPlace supports a comprehensive list of 14 constraint types as studied in Section 2.2.5. On the contrary, the implementation does not allow to define soft constraints which can be violated against some penalty factor (PF), e.g., in favor of overall savings. SLAs that are related to service performance are not discussed in the publication.

Finally, Hallawi et al. (2017) formulate a multi-dimensional vector bin packing problem in order to improve resource allocations in cloud environments. The authors developed two hybrid algorithms which combine a GA with a first-fit respectively next-fit heuristic. During evaluation, the hybrid algorithms outperformed pure heuristics found in the latest literature. Since the approach reduces the number of servers, the same are required to be homogeneous. Furthermore, multi-dimensionality concerns only the resource dimensions of items to be packed. Varying workload over time is not supported. The problem is subject to rather basic constraints: A service must be allocated to a single server and the

capacity requirements must not exceed the capacity limits of a server for any dimension. The consideration of performance was declared as future work.

2.4.2 Classification of the service placement problem

The related work, summarized in Section 2.4.1, spawns several variations of the SPP with different problem characteristics. In order to identify publications that are most closely related to the problem addressed in the thesis at hand, classification criteria were derived as part of the study. The respective scheme is depicted in Figure 2.14.

A fundamental characteristic concerns the state of services while solutions are deployed. In this regard, one can distinguish flexible environments which support frequent online relocations (live migrations) and those which benefit from rather stable designs which are optimized periodically in an offline manner. Online approaches are to be favored if load characteristics change frequently. Those approaches represent the interface to the research domain of load balancing. Processing time and network load, caused by the live migration (Hyser et al., 2007; Ankit et al., 2013) as well as the required number of migrations to reach the final state (Calcavecchia et al., 2012) are to be considered. Since high costs incurred by VM migrations prohibit an unlimited usage of this mechanism (Xu and Fortes, 2010), many of the online approaches formulate optimization problems which include to minimize the migration overhead. As pointed out by Hyser et al. (2007), online approaches sometimes require intermediate states that are worse than the initial state. The authors refer to this problem as iterative rearrangement problem. On the other hand, services whose load characteristics follow predictable patterns do not need to be rearranged continuously. Therefore, offline approaches benefit from a stable design that supports reliable operation on a long-term basis. Furthermore, performance degradations which result from the virtualization overhead, occurring, e.g., during live migrations, are not acceptable in certain domains. Offline approaches can start bin packing from scratch, i.e., calculating an optimal allocation that is to be deployed without considering the current or intermediate states. As a notable drawback, offline approaches require service downtimes which are to be planned in advance. Whether solutions to the SPP are deployed online or offline is generally limited by given infrastructure capabilities and, in turn, affects the type of applicable solution techniques.

While some of the studied publications use a sequence of demand values over time which forms the workload profile, others aggregate demands to a single average or peak value with no time dimension. The former is referred to as dynamic workload and the latter is classified as static workload, following the definition by Stillwell et al. (2010) and Feller et al. (2011). While the complexity of implemented algorithms is reduced significantly when eliminating the time dimension, static profiles do not allow to identify patterns, thus, parts of the optimization potential remains unexploited.

Beside the time dimension, additional dimensions of the problem are represented by the considered resource types. Many of the studied approaches use the CPU consumption as the major energy consumer to formulate problems with a single resource dimension.

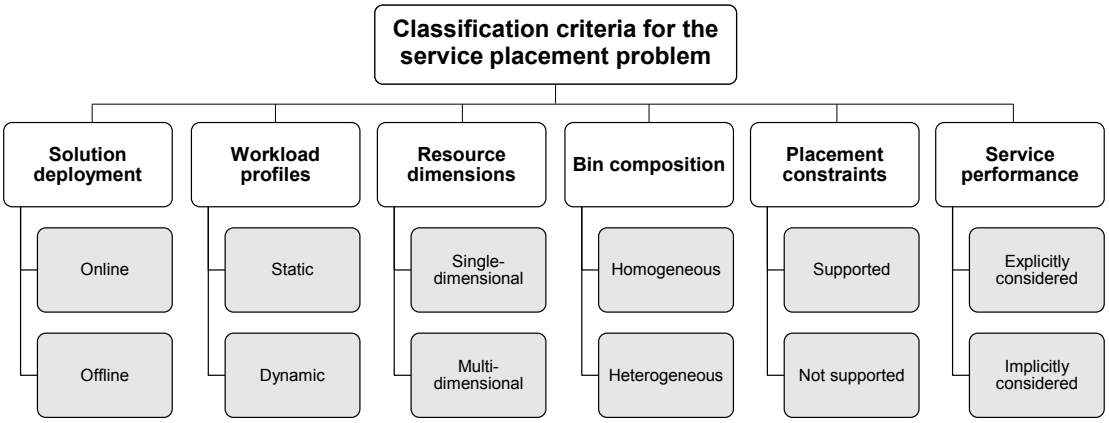

Figure 2.14: Classification of service placement problems.

If, in contrast, multiple resource types are considered by the solution technique, problem formulations are referred to as multidimensional.

Further restrictions to the application area are entailed by the assumed bin composition which can be either homogeneous or heterogeneous. If the formulated bin packing problem requires available servers to be equal, the approach is limited to homogeneous environments. In some case, servers are assumed to have equal types of CPUs but different numbers. These approaches must still be classified as homogeneous approaches since they hardly depict the reality of heterogeneous data centers with servers of different CPU architecture and frequencies (Petrucci et al., 2011).

As outlined in Section 2.2.5, placement constraints are inherent to many real-world problems, resulting from, e.g., compatibility issues, availability targets, or security policies. However, many publications disregard the existence of such restrictions and do not support to formulate individual constraints for a particular instantiation of the problem. The scheme in Figure 2.14 classifies problems according to this aspect.

Finally, the performance of relocated services within the computed design may be considered implicitly or explicitly. The majority of the studied publications carries out capacity management in a resource-centric manner as opposed to a user-centric manner. Therefore, SLAs which are related to performance are translated to technical metrics such as server utilization levels. Defined thresholds of these metrics are then supposed to ensure sufficient levels of performance. Approaches which employ this or similar strategies consider performance only implicitly since estimated levels of performance, typically expressed as throughput or response times, are not quantified.

Depending on the problem characteristics depicted in Figure 2.14, different techniques can be employed to solve the SPP. As pointed out by Stillwell et al. (2010), even in the case of a single resource dimension, several authors utilize heuristics to deal with the complexity of the problem. Heuristics efficiently address the bin packing problem and scale well with growing problem sizes. However, their flexibility is limited with respect to varying instantiations of the problem. For example, if placement constraints are to be considered, heuristics require to be adapted accordingly (Van et al., 2009). Entailing

implementations of pre- and/or post-processing steps may be costly if constraint types are subject to change.

2.4.3 Derivation of the research gap

In the following, each classification criteria of the SPP (cf. Figure 2.14) is discussed with respect to the related work and the application area of the thesis. As a result, a research gap is derived that is to be addressed by the designed artifact. The research gap serves as a basis for requirement elicitation in the subsequent chapter.

The state of services during solution deployment depends on service characteristics and technical capabilities of the environment in which the artifact is applied. The thesis at hand discusses an application areas in which services typically follow seasonal patterns (Speitkamp and Bichler, 2010; Bichler et al., 2006; Rolia et al., 2005; Setzer and Stage, 2010) and, therefore, do not need to be relocated frequently (Gmach et al., 2008). Using a maritime metaphor, frequent load changes are inherent to sport boats, e.g., services for data analyses which may be instantiated and distributed over existing resources on the basis of a frequently changing demand. In contrast, EAs represent heavy tanker vessels with a fixed but stable route. Their flexibility is limited in favor of reliability. In case of concurrency events, according to the marine law, sport boats are expected to dodge commercial tankers. Furthermore, as outlined in Section 2.1.1, performance degradations which result from live migrations and their technical overhead are to be avoided in the domain of EAs so that the existing infrastructure, in many cases, is geared towards the concept of application virtualization (cf. Section 2.2.2) and therefore limited to offline relocations.

The outlined seasonality, furthermore, allows to identify recurring workload profiles on a daily or weekly basis (Setzer and Stage, 2010). Many publications, in contrast, formulate a static variation of the SPP. According to ITIL, “the right level of capacity at the right time is critical [for successful capacity management]” (Hunnebeck et al., 2011, p. 159). Static approaches do not have a time dimension and provide always the same level of capacity. In contrast, dynamic variations of the SPP leverage existing saving potential by identifying complementary workload profiles.

Capacity, in many of the studied publications, refers to the CPU as the main energy consumer. Varasteh and Goudarzi (2017) identify a lack of applicability and efficiency in related approaches which, according to the authors, can be addressed by considering multiple system resources. Therefore, some approaches take further resource dimensions into account. According to a preliminary analysis of this thesis, CPU and memory demands do not correlate in the case of an EA (Müller and Bosse, 2016) and, therefore, must be considered independently. Otherwise, resource overloads may degrade performance and interfere successful execution of business functions in the computed design.

In addition, a major part (63%) of the studied solution strategies is limited to environments of homogeneous servers. According to the service design publication of ITIL, capacity management should include all kinds of technology for all IT components and

environments (Hunnebeck et al., 2011). In the domain of EAs, as defined in Section 2.1.1, applications typically comprise heterogeneous servers. Likewise, Petrucci et al. (2011) states that large clusters are formed by many heterogeneous machines with different capacities, number of CPU cores, frequencies, and specific devices. Therefore, homogeneous approaches can only be used to a limited extent.

Placement constraints are widely disregarded in the scientific literature although known to be mandatory for consolidation efforts (Speitkamp and Bichler, 2010; López-Pires and Báran, 2015; Pires and Barán, 2013). Similarly, Varasteh and Goudarzi (2017) conclude that constraints related to high availability of services are largely neglected by today's server consolidation techniques. Often, the requirement not to overload existing servers is formulated as the only constraint of the bin packing problem. Across the studied publications, few articles mention the importance to incorporate additional, selected constraints into the problem formulation but do not include the same into their evaluation activities. Only one of the proposed artifacts supports to define individual placement criteria on the basis of a number of supported constraint types (Hermenier et al., 2013).

Some of the studied approaches claim to be SLA-aware with respect to the service performance. However, it must be stated that non of the articles provides performance estimates in terms of response times although they are known to be the greatest issue from a user perspective. The vast majority of server consolidation approaches considers those higher level metrics only indirectly by mapping them to resource capacity values in a reasonable way (Stillwell et al., 2010). For example, Kusic et al. (2009) discuss response times resulting from SLAs but the authors only distinguish violated from satisfied SLAs. In fact, response times are not quantified and computed penalties do not depend on the degree of violation. Some approaches consider service performance implicitly by providing means to add spare capacity of up to 50% per service in order to ensure acceptable response times (Rolia et al., 2003, 2005). This strategy, in turn, lowers the addressable optimization potential to an extent that causes the entire consolidation effect to become questionable. The introduction of spare capacity is certainly a valid instrument to catch unpredictable load peaks but should not be overused to compensate uncertainty of service performance. Therefore, current approaches lack an explicit evaluation of identified solutions with respect to expected response times or throughput.

To summarize, if multiple resources are considered by the related work, their workload profiles are often static. On the other hand, approaches that support dynamic workload profiles in most cases consider only one resource dimension. Only exceptions to this rule represent the publications of Wood et al. (2007), Gmach et al. (2008), and Feller et al. (2011). However, the latter two approaches are limited to homogeneous servers and non of the three supports placement constraints. Only one of the proposed solution techniques allows to define placement constraints (Hermenier et al., 2013) as part of the problem instantiation (cf. Section 2.2.5). The authors, however, formulate a static SPP which is limited to a homogenous composition of bins. In addition, in the studied articles, service performance is either disregarded or implicitly considered by mapping performance metrics

to certain levels of resource utilization. Expected levels of response times or throughput are not quantified in any of the articles although required to map the quality and costs of solutions to the targeted quality of service performance as defined in the SLAs.

Therefore, a multidimensional SPP which leverages dynamic workload profiles while considering individual sets of placement constraints is desirable. Additionally, the SPP must explicitly provide expected levels of performance for the computed solution candidates in order to allow for their evaluation with respect to given SLAs. Suitable solution techniques to solve the SPP in an offline manner are to be identified on the basis of the outlined problem characteristics.

2.4.4 Summary

Related work was identified on the basis of the VMP literature review carried out by López-Pires and Báran (2015) which was complemented by a subsequent forward-backward search. While the problem of VMP has been worked on by a variety of publications, the maximization of resource utilization and the maximization of performance represent the least studied objective functions. The problem, in this thesis, is formulated broadly as a service placement problem which may be classified according to the service state during solution deployment (online or offline), characteristics of the workload profile (static or dynamic), the number of resource dimensions (single- or multidimensional), the bin composition (homogeneous or heterogeneous), and finally, the consideration of placement constraints (supported or not supported) and service performance (implicitly or explicitly considered). To the best of the author's knowledge, no related article satisfies all classification criteria with respect to requirements of the targeted domain of enterprise applications.

2.5 Summary of the state of the art

Enterprise applications have exceptional requirements on performance due to their indispensability for successful business execution. The process of capacity management is carried out with the objective to satisfy targeted levels of performance in alignment with given service level agreements. Since, on the other hand, costs are to be minimized as part of the process, data center operators seek to increase energy efficiency, e.g., by avoiding idle resources. However, according to several recent studies, mean server utilization typically varies between 10-20%. Resulting optimization potential can be addressed by means of server consolidation. The respective problem is NP-hard and formulations vary across related work. Solution algorithms include exact approaches, heuristics and metaheuristics. With respect to realities of the given domain, existing artifacts lack to consider multiple resource dimensions of heterogeneous servers, dynamic workload profiles, placement constraints and service performance at the same time. The latter may be estimated by means of model-based, measurement-based or machine learning-based techniques on a transactional basis. Existing limitations of either technique may be overcome if a vast amount of

performance counters was shared within the domain of an enterprise application type and used to train machine learning models. This idea is abetted by the fact that the majority of enterprise applications implement customized versions of a commercial-off-the-shelf software whose usage is largely dominated by standard business transactions. Model types which are known to deal well with high dimensionality and non-linear relations are, for example, Random forests, Support vector machines, and Boosted trees. The described prediction technique enables to evaluate performance of managed applications already in the design phase without the need to implement a running prototype. Therefore, it may be applicable in order to evaluate solution candidates to the service placement problem with respect to the expected SLA compliance.

3 Design of the performance prediction supported service placement (PPSS)

This chapter covers the design of the research artifact which represents a method, termed *Performance prediction supported service placement (PPSS)*. Design requirements are elicited and a variation of the SPP is formulated accordingly. Since the designed method includes an evaluation of solution candidates to the SPP with respect to performance-related SLAs, one section is dedicated to corresponding prediction techniques and their validation. This chapter therefore takes the perspective of a capacity management provider who maintains solution algorithms and validates prediction models before the same can be put into use. Usability from the perspective of a capacity management consumer, in contrast, is demonstrated as part of the subsequent evaluation.

3.1 Conceptual design

Following the initial requirement elicitation, the conceptual design is demonstrated by means of a process model which depicts the designed method. A central component of the process model is a knowledge base for application performance monitoring data, which is introduced at the end of this section.

3.1.1 Design requirements

The design science research guidelines (cf. Section 1.2) require to satisfy laws of the problem domain which cannot be controlled. On the one hand, these laws were identified from the literature by the study of related approaches and the application area. The result culminated in the research gap (cf. Section 2.4.3). On the other hand, industry representatives contribute practical requirements that must be considered in the artifact's design in order to ensure practical usability. Accordingly, requirements engineering processes are often modeled as variants of the main activities elicitation, specification, and validation while both the problem domain and the users represent sources of relevant input (cf. Figure 3.1). Hence, the following requirement specification builds upon characteristics of the problem domain as introduced in Chapter 2 and upon user input from a number of expert interviews with representatives of an industry partner who plans to apply the artifact. First, functional requirements F1 to F9 are introduced. Quality requirements follow and are labeled Q1 to Q4. By the end of this subsection, all requirements are summarized in Table 3.1.

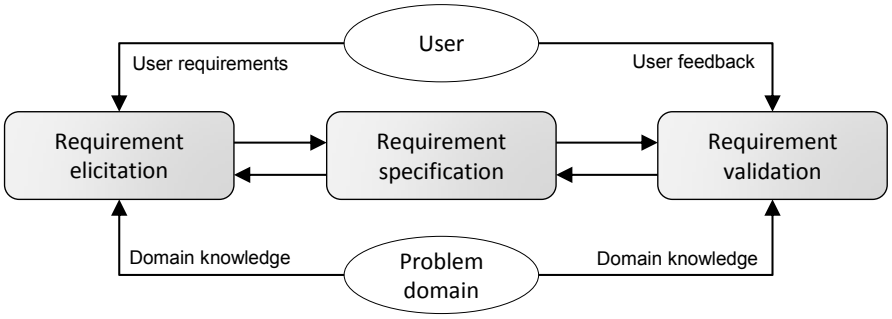

Figure 3.1: Requirements engineering process, based on Ebert (2014).

The artifact minimizes required server capacity (F1). It is the goal of the capacity management process to balance costs and performance. Cost savings can be achieved by means of server consolidation techniques as outlined, e.g., in Section 2.2.1. The extent of savings depends on the amount of capacity that can be eliminated along with associated energy costs, power and cooling costs, as well as personnel costs. Therefore, it is the main requirement of the artifact to minimize server capacity.

The artifact minimizes SLA violations (F2). Server consolidation implies the risk to overload single servers if utilization levels were raised too ambitiously. In this case, performance may be degraded and related SLAs are violated. Therefore, besides above listed types of operations costs, another portion of expenses is caused by penalties which are invoked on SLA violation. The number and degree of SLA violations is to be minimized in order to keep the total costs of the computed design low (Hyser et al., 2007). While SLAs which restrict the placement (such as availability or security agreements) may be modeled as placement constraints, compliance with performance-related SLAs is assessed by means of techniques covered in function requirement F8.

The artifact follows an offline optimization approach (F3). As carved out in Section 2.4.3 on the research gap, EA workload profiles follow seasonal patterns and, therefore, allow to be optimized periodically instead of continuously. Many data centers employ application virtualization techniques respectively which support to relocate services in a way that the application is stopped on a source server and started on a target server. This offline approach requires to plan a downtime for the services under optimization. Online approaches, on the other hand, must consider given SLAs and placement constraints also during the relocation phase. If the added complexity is left unaddressed by the problem formulation, temporary but unplanned violations may interfere business operations. Considering the characteristics of the problem domain, the artifact is therefore designed to optimize service allocation in an offline state.

The artifact utilizes dynamic workload profiles (F4). Static workload profiles do not allow to identify existing load patterns (e.g., daily or weekly recurrences), thus, optimization potential is left unaddressed (cf. Section 2.4.2 and Section 2.4.3). Hence, the optimization problem formulation must incorporate a time dimension which represents dynamic workload profiles.

The artifact supports heterogeneous server environments (F5). According to the definition of EAs (cf. Section 2.1.1), applications are typically hosted on heterogeneous servers. To comply with the target domain, the artifact must support to consolidate servers of any type and capacity.

The artifact considers CPU and memory resources (F6). As outlined in Section 2.4.3, multiple resource dimensions are to be considered in order to avoid overloads of neglected resource types. For example, a preparatory analysis of workload profiles from 364 running EA services revealed no linear correlation between their CPU and memory demands over time (Müller and Bosse, 2016). Hence, the artifact must allow to formulate and solve problems with multiple resource types. To pay tribute to the increasing importance of in-memory computing in the domain of EA, CPU and main memory are used as the two most important resource dimensions for the sake of demonstration.

The artifact considers placement constraints (F7). In ITIL, it is excessively emphasized that capacity management needs to be aligned with business needs and, therefore, must comply with resulting requirements on availability, security, or licensing. A list of constraints was identified from the literature and provided in Section 2.2.5. As part of the artifact's design, this collection is to be checked for completeness with respect to the practical applicability. Resulting types of constraints must be considered by the artifact when supporting placement decisions.

The artifact predicts transactional performance using machine learning (F8). In the domain of EAs, performance is of exceptional significance (cf. Section 2.1.1). As a consequence, solution candidates must undergo a performance evaluation which takes expected response times on a transactional basis into account. Different means exist to estimate performance. According to ITIL, a key success factor for managing capacity is its earliest possible application, that is already during the design stage. Any quality of service which is added in later stages is associated with higher correction costs (Hunnebeck et al., 2011). As stated in Section 2.3.2, machine learning-based techniques can leverage the potential of shared performance counters and, this way, generalize well to unseen data already in the design stage. The concept relies on a massive amount of performance metrics which are comparable within a defined domain. Since EAs are mainly based on COTS functionality (cf. Section 2.1.1) and utilize instrumentation techniques for continuous monitoring (cf. Section 2.1.4), the concept may be applied to the APM output of various

EAs within the domain of a particular type of COTS software. In fact, ITIL recommends to identify customers of the same product in order to learn about resource implications from them (Hunnebeck et al., 2011, p. 174). This recommendation is addressed by the artifact while the time-consuming process of knowledge engineering is automated through machine learning. Since performance, from an end-user perspective, is expressed as response time and accordingly formulated in SLAs, prediction models must estimate response times of a business transaction. Predicted values, in turn, are used to meet the functional requirement F2.

The artifact provides its function as a service (F9). Performance counters hold valuable information about the measured application and expose ways for improvement (Browne et al., 2000). Consequentially, the simplification of hardware counter collection was addressed by Browne et al. who propose a cross-platform infrastructure which standardizes the names for the metrics. However, subsequent data analysis still remains the greatest challenge and the use of artificial intelligence is often blocked by organizational barriers. In a recent Gartner survey, 196 organizations were asked to rate their maturity with respect to data and analytics. Out of six levels, 60% of the organizations rated themselves in the lowest three levels which range from basic ad hoc data analysis to a systematic analysis of data from exogenous sources. In general, traditional forms of data analysis dominate machine learning techniques which is not expected to change in the near future. Greatest barriers include the generation of value from the analysis (Gartner, 2018a). Therefore, the process of n-n cross-organizational learning in the field of capacity management is centralized in order to simplify both the sharing of performance counters and the subsequent learning phase. This way, challenging tasks of data analysis are outsourced to a capacity management provider who releases domain-specific performance models that have been trained on the basis of performance counters from all environments attached to the service. Such models are used to evaluate solution candidates to the SPP. Its solution process is offered as a service to a capacity management consumer who, on the other hand, stays focused on his core business.

The artifact employs capacity savings which are of economical relevance (Q1). The amount of savings that can be achieved depends on the current utilization levels and workload characteristics. The authors of Speitkamp and Bichler (2010) formulated a related problem which, similar to this thesis, considers dynamic workload profiles and was evaluated using services from the SAP domain. The authors achieved average savings of up to 31% server capacity when they consolidated services on the basis of daily workload patterns using a single resource dimension and no placement constraints. Therefore, the artifact is expected to reveal, in average, not less than 30% savings in unconstrained scenarios. For constrained scenarios, no related measures could be found in the literature since the combination of modeled constraints is highly individual for each problem instantiation. However, the existence of constraints is expected to lower the savings, hence,

average savings of at least 20% are to be achieved across any scenario so that the practical application of the artifact is economically noteworthy.

The prediction error is within an acceptable range (Q2). Boundaries of acceptable errors depend highly on the problem domain and must be defined by the modeler (Almeida, 2002). In the domain of capacity management, accuracies from 10 to 30% are typically acceptable (Almeida and Menascé, 2002; Menascé et al., 2004). This range is consistent with published results on software performance modeling, e.g., in Rathfelder et al. (2014) and Bontempi and Kruijtzer (2002). Therefore, the mean absolute percent error is expected to be below 30% when predicting the performance of a business transaction. Furthermore, the RMSE of a model must fall below the standard deviation of the target value in order to justify modeling effort as opposed to simply computing the average value of the measured target.

The artifact must be well scalable for very large environments (Q3). As stated in Section 2.2.1 and further documented in Section 2.4.1, today's data centers may be formed by hundreds and thousands of servers. Such large problem sizes, depending on the solution technique, may increase computation time to the magnitude of minutes and hours (e.g., in Speitkamp and Bichler (2010)). However, due to the form of service delivery, determined by functional requirement F9, solution candidates are to be identified within seconds. This way, the capacity management consumer is able to interact with a user interface and perform multiple solution iterations with varying parameters, e.g., refined constraints.

Label	Specification summary
	Functional requirements
F1	The artifact minimizes required server capacity.
F2	The artifact minimizes SLA violations.
F3	The artifact follows an offline optimization approach.
F4	The artifact utilizes dynamic workload profiles.
F5	The artifact supports heterogeneous server environments.
F6	The artifact considers CPU and memory resources.
F7	The artifact considers placement constraints.
F8	The artifact predicts transactional performance using machine learning.
F9	The artifact provides its function as a service.
	Quality requirements
Q1	The artifact employs capacity savings which are of economical relevance.
Q2	The artifact predicts within acceptable error ranges.
Q3	The artifact must be well scalable for very large environments.
Q4	The artifact must be well maintainable and customizable.

Table 3.1: Requirement specification.

The artifact must be well maintainable and customizable (Q4). The individual business input in the form of workload profiles and individual constraints is hardly generalizable. Furthermore, new environments potentially entail additional requirements which were not considered in previous problem formulations. For example, Hermenier et al. (2013) state that new uses of data centers continuously trigger new placement constraints to arise. Therefore, the artifact must be designed to be flexible and extendable with respect to individual constraint types.

Requirement validation requires to get feedback from the problem domain and from the potential users. While the former was collected as part of scientific reviews and conference attendances, the latter was discussed in a number of workshops with domain experts which guided the process to the evaluated requirement specification. Table 3.1 summarizes the functional and quality requirements. Based on this specification, design decisions are made throughout the subsequent sections. Furthermore, the requirements serve as a baseline during the evaluation of the artifact.

3.1.2 Process model

The artifact designed in this work represents a method for capacity management. Therefore, a process model helps to illustrate the artifact and its sequence of tasks. The process was modeled on the basis of preliminary work by Almeida (2002) (cf. Figure 2.4) but, according to the requirement specification and the hypothesis of this work, incorporates tasks from the domain of data mining. Therefore, data mining steps, given by the CRISP-DM process (cf. Figure 2.12), are used within the method in order to ensure the accurate application of machine learning modules. Furthermore, while Almeida focuses on the capacity management of web services, modifications were made in order to comply with the special characteristics of the EA domain. Figure 3.2 illustrates the designed method using the Business process modeling notation (BPMN) in its current specification.

A previous version of the process was published and evaluated in Müller et al. (2017b). As required by F9, the artifact must provide its function as a service. Accordingly, the concept of BPMN swimlanes is used to distinguish tasks of n capacity management consumers from tasks of one capacity management provider. Data exchange is realized by means of interfaces to an application performance monitoring knowledge base (APM-KB) which is hosted by the provider. The concept of a central knowledge base is inspired by the MAPE-K (Monitor-Analyze-Plan-Execute-Knowledge) feedback loop, published by IBM in 2005, where a monitored system is to be improved in a number of iterations (IBM, 2005). The APM-KB stores measured data, including performance counters, which is used by the provider to train performance models and by the consumer to extract workload profiles and server capacity values that form the input to the SPP. Section 3.1.3 is dedicated to the structure of the APM-KB. Measurement data results from APM activities which may be carried out continuously or periodically for a given EA environment at consumer site. A capacity change request triggers subsequent steps to manage capacity; it may arise in the context of upper-level business changes or, e.g., as part of consolidation projects. The

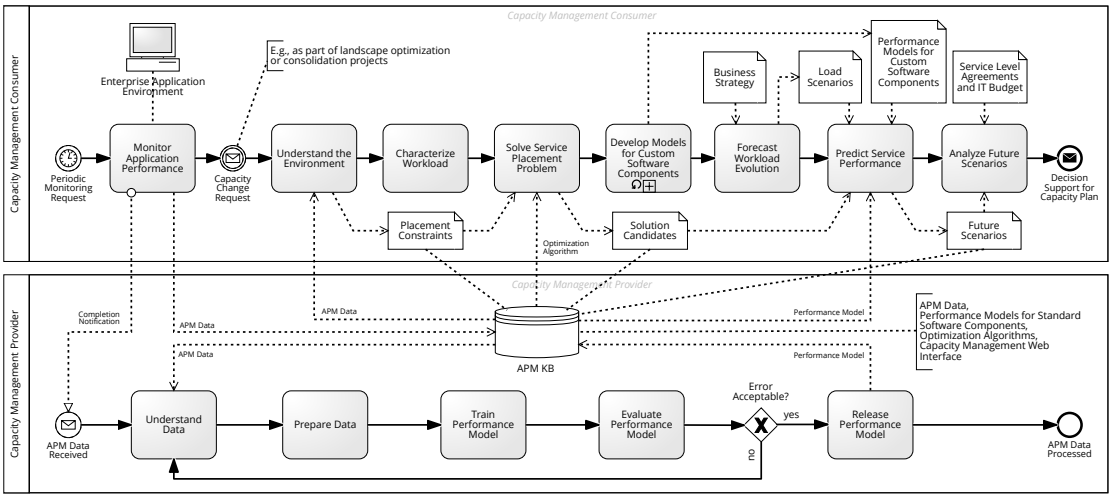

Figure 3.2: Capacity management according to the PPSS method.

process steps, introduced in the following, are supported by (graphical) user interfaces which are hosted as part of the APM-KB.

The first task in both traditional capacity management processes and data mining processes is always to gain a business understanding. Therefore, the step ***Understand the environment*** entails a descriptive analysis of running services and servers. Here, current server utilization levels are analyzed in order to obtain the optimization potential. Furthermore, given constraints and SLAs that limit the degrees of freedom for optimization are to be identified.

The next task, ***Characterize workloads*** involves to build workload profiles for each running service from the monitored data. As argued in F4, the seasonality of EA load patterns allows to group values on a daily or weekly basis. Typically, peak values of a given time interval are to be used here, e.g., the maximum resource demands per hour. Alternatively, Speitkamp and Bichler (2010) demonstrate that the usage of the 95th percentile for each workload profile increases the optimization potential. However, alike strategies may foster overloads and are, as a last resort, subject to the risk attitude of the decision maker. Another outcome of this task is to identify frequently used transactions which are of interest for later performance evaluation.

Having workload profiles and capacity limits in place, an SPP can be instantiated and solution candidates which consider identified placement constraints are to be found. The step ***Solve service placement problem*** utilizes solution algorithms, maintained by the capacity management provider, and feeds the algorithms with the individual data of the consumer's environment. The solution process may be iterated using different parameters (that is varying placement constraints or algorithm parameters) until suitable solution candidates have been identified. This process step is designed in-depths in Section 3.2 along with the problem formulation, the solution fitness definition and the algorithm instantiation. Solution candidates are stored in the APM-KB for further analysis.

The subsequent challenge is to evaluate solution candidates with respect to their

total costs. Particularly, the method seeks to minimize the operations costs and the SLA penalty costs which are invoked when performance levels are violated for a business transaction of interest. For this purpose, performance models are used to estimate mean transactional response times, expected from each solution candidate. For example, one solution candidate may increase mean server utilization up to a level which eliminates a significant share of servers but, on the other hand, causes response times to exceed defined service levels in the majority of observations. This solution would be favored if performance was neglected although it may turn out to be a rather expensive option. In Section 2.1.1, it was documented that standard functionality dominates custom transactions. Therefore, in the majority of cases, performance models can be applied that were constructed by the capacity management provider on the basis of APM data which is shared in the knowledge base. The process of training the standard models is covered by the lower swimlane and is going to be explained later in this section. However, in case custom transaction performance is to be evaluated, individual models must be constructed. This task is covered by the subprocess ***Develop models for custom software components***. It is out of scope in this work and therefore collapsed in Figure 3.2. However, model-based techniques using simulation engines or analytical solvers, as summarized in Table 2.2, may be applied.

As a basis for performance estimation, historical load, as found in the workload profiles, could be assumed in the future. Nevertheless, capacity management must consider future business plans and requirements as indicated by the service portfolio (Hunnebeck et al., 2011). Hence, alternative scenarios may result from the *business strategy*, e.g., a pending product release which is expected to increase load in the near future by a certain factor. Alike forecasts are subjcet to the step ***Forecast workload evolution*** and involve communication with business units in order to assess their impact on load related metrics (Cherkasova and Rolia, 2006). As a result, different *load scenarios* represent alternative business developments in whose regard solution candidates are to be evaluated.

The actual performance projection is carried out as part of the subsequent task ***Predict service performance***. Here, the identified load scenarios are applied on each solution candidate, resulting in a number of different input features which contain varying load and hardware characteristics. Performance models, either standard or custom ones, are used to estimate expected response times of business transactions. The result are *future scenarios*, each representing a combination of load scenario and solution candidate along with predicted levels of transactional performance. Besides the configured load factor, an important parameter to control the number of SLA violations is the maximum allowed server utilization level which was defined as a placement constraint. In general, solution candidates which allow higher server utilization come with increased risk of performance degradations (Beloglazov et al., 2012).

The identified future scenarios, in the final step, are to be analyzed with respect to their total costs while *Service level agreements and IT budget* must be considered. As argued by Hyser et al. (2007), a system that understands both power costs and penalty

costs may choose to violate service levels if the financial loss was less than expenses required to meet the service level. This trade-off is to be solved as part of the step ***Analyze future scenarios*** which leads to decision support, formulated in a capacity plan. The capacity plan, according to ITIL, outlines scenarios and forecasts for the future capacity demands; furthermore, it recommends actions to address those demands. Recommended actions must be associated with cost estimates (Hunnebeck et al., 2011). As this plan serves as the basis for subsequent decision making, it represents the end of the designed capacity management process.

Whenever one of the capacity management consumers piles a certain amount of new APM data, it is uploaded to the APM-KB via a defined interface. The data is then used by the capacity management provider to update existing performance models or to construct new models. Therefore, a sub-process on the basis of the CRISP-DM methodology is followed, modeled in the lower swimlane of Figure 3.2. Here, as part of the steps ***Understand data*** and ***Prepare data***, cleaning and filtering techniques are applied. Depending on the type of the model, features are to be normalized and selected. The task ***Train prediction model*** utilizes machine learning techniques on the basis of shared performance counters as introduced in Section 2.3.2. This way, models learn from observations which are not limited to a single consumer environment but cover a massive number of different hardware, software and workload characteristics. Consequently, from a consumer perspective, the number of applicable what-if simulations is extended to unseen scenarios. If, as part of the step ***Evaluate performance model***, acceptable error metrics were computed, trained models can be ***released*** for application. The range of acceptance is defined by quality requirement Q2. In the contrary case, model accuracy may be tuned in further iterations of the sub-process.

As ITIL recommends to store outcome from capacity management sub-processes in a CMIS (cf. Section 2.1.3), all artifacts which arise in the course of the process (that is optimization algorithms, placement constraints, solution candidates, load scenarios, performance models, and future scenarios) are stored in the APM-KB. This way, comprehensibility and repeatability is ensured. Furthermore, results and their complete solution path is transparent in case further analysis is desired.

3.1.3 Application performance monitoring knowledge base

A central component of the designed method is the APM-KB, as illustrated in the process model of Figure 3.2. As pointed out by Matsunaga and Fortes (2010), the collection of as many run-time attributes as available can be beneficial to feat learning algorithms and, this way, to make predictions. As a matter of fact, the integration of data is the basis for a data-driven capacity management. In this regard, the APM-KB takes the role of the CMIS and, according to ITIL, holds the following types of data:

- **Service data:** Measured data, including workload volumes, transaction rates, and performance metrics are to be stored in the APM-KB. The data is used for model training and to identify workload profiles as input to the SPP.

- **Component utilization data:** This data type refers to current utilization levels for each resource type and their capacity limits. For example, each server may have a capacity limit for its CPU and for its main memory.

- **Business data:** Here, load scenarios which result from the business strategy are to be stored. The actual business strategy, of course, is subject to data privacy and must not be shared with the capacity management provider.

- **Financial data:** In the final capacity plan, each solution candidate is associated with costs. Therefore, operations costs and SLAs with their respective penalties are to be stored in the APM-KB.

ITIL further documents that access to the CMIS is often provided by a web interface which offers different views on the data. To accomplish all tasks, introduced in the preceding Section, the APM-KB is formed by the following layers:

- Presentation: A web interface supports tasks of the process and visualizes data.
- Analysis: A statistics server provides machine learning and optimization functions.
- Data: A relational database ensures consistency and access control.

Since the name and the format of relevant metrics varies for each type of COTS EA (Chen et al., 2002), different schemata may be maintained on the data layer. In the present work, e.g., a schema was created to hold statistical records of SAP EAs. The data layer may be further filled with publicly available information. In the domain of SAP EAs, aggregated measurement results are published by performance analysts, e.g., in Kowarschick (2019). Here, results of the SD benchmark were used to compute the CPU capacity of different server types in SAPS (cf. Section 2.1.1). Those values may be used to update the APM-KB with server capacity values and, in turn, update models with newest configurations without the need to implement respective configurations for each organization individually.

3.1.4 Summary

The artifact, designed in this work, involves two types of actors: One capacity management provider offers standard performance models and solution techniques for the SPP to multiple capacity management consumers. The concept enables consumers to learn from characteristics of other environments and apply the extracted knowledge during own capacity management exercises. This way, cost-effective performance evaluation of solution candidates is enabled, with the objective to minimize the sum of expected operations costs and SLA penalty costs. To apply the method successfully, it is linked to a number of functional and non-functional requirements that are to be fulfilled.

3.2 Multi-dimensional service placement

The SPP is to be formulated and solved according to the specified requirements. Therefore, this section deals with techniques to identify solution candidates while the subsequent section introduces means to evaluate their performance.

3.2.1 Problem formulation

According to the classification provided by López-Pires and Báran (2015), the functional requirements allow to formulate a mono-objective optimization problem with multiple resource dimensions and one time-dimension. The objective is to minimize needed server capacity or, in other words, to maximize unneeded server capacity. As stated in Section 2.2.3, the problem is closely related to combinatorial bin packing. The most related formulation was provided by Speitkamp and Bichler (2010). On this basis, the SPP is formulated as $SPP = (T, H, I, C, c_o, c_{sla})$ with

- $T \in \mathbb{N}_+$ the number of intervals which form the time dimension, e.g. $T = 24$ for a daily season;
- $H = \{h_1, \ldots, h_n\}$ the set of servers or hosts with $h_j : R \to \mathbb{R}_+, 1 \leq j \leq n$ defining the capacities for $R = \{1, \ldots, k\}, k \in \mathbb{N}_+$ resource types;
- $I = \{i_1, \ldots, i_m\}$ the set of services or items with $i_j : R \times T \to \mathbb{R}_+, 1 \leq j \leq m$ defining the resource demands (workload) for R resource types in T time intervals of the season;
- $C = \{C_1, \ldots, C_k\}$ a set of k placement constraints, comprising the main resource constraint C_{res}, the maximum server utilization constraint C_{util}, and an arbitrary number of further constraints;
- $c_o : (I \to H) \to \mathbb{R}_+$ a cost function indicating the operations costs, in terms of required server capacity, for a season for an allocation from services to servers; and
- $c_{sla} : (I \to H) \to \mathbb{R}_+$ a cost function indicating the penalty costs, issued from SLA violations, for a season for an allocation from services to servers.

In a first step, it is the objective to identify an allocation $a : I \to H$ that minimizes the cost function c_o subject to the constraints of C. Here, C_{util} defines the optimization pressure (also referred to as consolidation pressure), depending on the risk attitude of the decision maker. While many of the related approaches pack servers up to their limit of 100%, some cases may require to manage capacity more conservatively. In the domain of SAP, e.g., mean server utilization levels around 65% are generally considered to fairly balance the trade-off between costs and performance (Janssen and Marquard, 2007). Therefore, different values for C_{util} yield alternative solution candidates which represent different risk attitudes. Among this set s of solution candidates, it is the objective of a secondary problem to identify an allocation that minimizes the sum of the cost functions

c_o and c_{sla} in order to identify a Pareto-optimal solution with respect to the total costs. For this purpose, solution candidates must undergo performance projections as will be explained in Section 3.3. The primary problem is addressed in the following.

3.2.2 Solution fitness

The quality of a solution is evaluated using a fitness function which incorporates the operations costs c_o for an allocation a. Costs are expressed by the amount of used server capacity and, accordingly, can be measured in SAPS. For simplicity, the sign is converted, thus, $-c_o(a)$ is to be maximized. Since constraint violations lower the solution quality, fitness is to be reduced, or penalized, depending on the degree of constraint violation. The main constraint C_{res} of the SPP is to avoid resource overloads. Therefore, a penalty factor $p_{res}(a)$ for resource overloads is introduced. Another penalty factor $p_c(a)$ represents violations of additional placement constraints C. Hence, the fitness function $f(a)$ for an allocation, as expressed in Equation 3.1, is to be maximized:

$$f(a) = -c_o(a) - p_{res}(a) - p_c(a) \tag{3.1}$$

If the sum of all instances' resource demand, placed on one server, exceeds the provided server capacity at a time, resource constraint penalization is invoked. Since the SPP supports the consideration of multiple resource types, those must be normalized when the penalty factor $p_{res}(a)$ is computed. For this reason, scaling factors s_r for each resource type ensure that any kind of resource overflow penalizes the fitness equally. Furthermore, the degree of overflow is critical. While minor overflows may be acceptable, large overflows entail severe consequences such as significant performance degradations or service unavailability. In order to tolerate little overflows and penalize large ones, penalty factors are scaled according to a near-feasibility threshold (NFT) and subsequently squared, as proposed by Kulturel-Konak et al. (2003). For each resource type, the NFT must be set to a threshold that is barely acceptable. In simple terms, subsequent squaring entails that penalty factors below the NFT are reduced while values above the NFT are increased. Alternatively, exponentiation greater than quadratic reinforces the effect. The outlined aspects are expressed in Equation 3.2:

$$p_{res}(a) = \sum_{r \in R} s_r \cdot \sum_{t \in T} \sum_{h \in H} \left(\frac{\max\left(0, \left(\sum_{i \in I, a(i)=h} i(r,t)\right) - h(r)\right)}{NFT_r} \right)^2 \tag{3.2}$$

If resource overflows are present, the numerator of Equation 3.2 becomes positive and penalization is invoked. In all other cases, the numerator equals 0 and, divided by any NFT, remains 0. Penalization for the violation of all additional placement constraints is realized by means of $p_c(a)$ which is defined by the sum of all penalty factors for each constraint type:

$$p_c(a) = \sum_{co \in C} pf(co) \tag{3.3}$$

According to Coit and Smith (1996), it is important to define the degree of penalty depending on the severity of the violation. Therefore, penalty factors must be defined for each constraint type individually and may vary across different SPP instantiations depending on individual weightings provided by the capacity management consumer. For example, a consumer may define that the central production service must never be placed together with its backup service as obligatory condition while some development services should preferably but not necessarily run isolated from each other. The types of placement constraints, currently designed in the SPP, are introduced in the following section.

3.2.3 Placement constraints

"A design artifact is complete and effective when it satisfies the requirements and constraints of the problem it was meant to solve" (Hevner et al., 2004, p. 85). Here, besides the resource constraint, introduced in the preceding section, a number of individual placement constraints limit the solution space of the SPP. In particular, the ongoing trend to reduce costs by outsourcing service operations to large-scale data centers (Boloor et al., 2010a) emphasizes the need to incorporate placement constraints, e.g., in order to isolate services of different customers from each other. A review of placement constraints as existing in the related literature was conducted in Section 2.2.5. To evaluate the relevance of placement constraints within the SPP also from a practical point of view, two types of semi-structured expert interviews were conducted with industry representatives. This method was chosen as experts are known to take positions which may differ from research and to provide aspects which become relevant if artifacts are to be applied in practice.

The first interview[1] was held together with a focus group of a global IT company who plans to become a capacity management provider, according to the process model in Figure 3.2. It was the goal of this focus group to evaluate the practical utility of each placement constraint found in the literature. The focus group comprised three business and IT consultants of the service provider, each having more than 20 years of experiences with customer concerns, mainly in the domain of SAP application and database operations. In addition, the author of this work and a co-researcher, who presented parts of the interview results in his master thesis (Akhras, 2017), participated. As a result of the interview, practical utility for each constraint type was rated. Another outcome is that, depending on the individual preferences of the consumer, some constraints may represent *hard* conditions while others may represent rather *soft* requests which are nice to have. Finally, experts concluded that constraints may apply to single services or servers or, in contrast, to a group of services or servers of the same type. Therefore, all participants confirmed that a classification of services and servers may ease constraint modeling effort.

The second type of interview[2] was held twice together with two different pilot users who took the role of capacity management consumers. This type of interview aimed at evaluating the feasibility and completeness of the designed set of placement constraint

[1] Expert interview at September 20, 2016
[2] Expert interviews at February 07, 2018 and February 14, 2018

ID	Placement contraint type	Default penalty factor
1	Resource constraint	Sum of relative overflows for all resource types, referring to the server capacity limits (cf. Section 3.2.2)
2	Specific location constraint	Ratio between the number of incorrectly located services and the number of services to be located
3	Anti-location constraint	Ratio between the number of incorrectly located services and the number of services to be anti-located
4	Colocation constraint	Ratio between the number of not colocated services and the number of services to be colocated minus 1
5	Anti-colocation constraint	Ratio between the number of falsely colocated services and the number of services to be anti-colocated minus 1
6	Maximum number of services constraint	Ratio between the number of servers for which maximum service number is exceeded and the number of servers in the constraint
7	No neighbors constraint	Ratio between the number of non-isolated services and the number of services to be isolated
8	Maximum utilization constraint	Sum of relative overflows for all resource types, referring to the maximum utilization limits
9	Extra resource constraint	Sum of relative overflows for all resource types, using the redefined resource demands
10	Component exclusion constraint	Components are excluded before problem instantiation, hence, penalization is not needed

Table 3.2: Designed placement constraint types.

types. Here, two workshops were conducted together with five respectively three representatives of two companies who participated in a pilot phase as part of the artifact's evaluation: A large German retail company, operating more than 10.000 points of sale with a yearly revenue of more than 40 billion Euro, and, secondly, a large German insurance company with a yearly revenue of more than 700 million Euro. Both companies operate their IT infrastructure fully on-premises and, therefore, continuously seek to optimize cost-effectiveness. In both cases, COTS EAs of SAP are operated in an application-virtualized environment. The participants tested parts of the designed process for six weeks prior to the workshop. The main focus was put onto the task *Solve service placement problem* (cf. Figure 3.2). Results are presented in more detail as part of EVAL 4 in chapter 4, however, two of the questions are relevant already during the design stage of the artifact. First, participants were generally asked to rate the usefulness of the possibility to define placement constraints for the optimization task. Each participant rated on a scale from 0

(irrelavant) to 10 (essential). Results, grouped by organization, account to 9 respectively 10. Second, participants were asked if any placement restrictions could not be modeled using the provided types of placement constraints. As a result, the retail company claimed an additional constraint type which allows to specify a set of services that can be relocated at the same time. The requirement was explained by the fact that each service has defined time slots in which a downtime, required to deploy solutions, can be realized. These slots may not overlap for all services so that it would be desirable to limit the SPP to a defined set of services. As a consequence, an additional constraint type was designed which is termed *Component exclusion constraint*. It allows to specify services and servers that must be excluded from the automated problem instantiation in order to limit the problem to services which can be relocated in parallel.

To summarize the results of the literature review and the expert interviews, Table 3.2 lists all constraint types which are supported by the SPP. The constraint types are sorted by utility in decreasing order on the basis of the ratings provided in the first expert interview. For each constraint type, individual penalty factors can be defined (cf. Section 3.2.2) in order to model hard and soft constraints as requested by the interviewed industry experts. Therefore, Table 3.2 lists default penalty factors which were derived during sensitivity analysis but may be subject to individual parametrization by the consumer.

3.2.4 Solution algorithms

To solve variants of the SPP, heuristics and metaheuristics dominate the literature (López-Pires and Báran, 2015; Varasteh and Goudarzi, 2017). Exact solution techniques, in contrast, are rarely applied due to their lack of scalability with increasing problem sizes. To comply with quality requirement Q3, exact solution techniques were not considered in the present work either. While heuristics are known to solve bin packing problems highly efficient, metaheuristics are more flexible because they do not require specific problem knowledge. In order to evaluate different solution techniques with respect to their handling of the problem characteristics, the following eight algorithms were selected to solve the SPP:

- First-fit decreasing (FFDmax and FFDsum): A heuristic which sorts the list of services by their resource demands in descending order before each placement. FFDsum sorts on the basis of accumulated demands over time while FFDmax sorts on the basis of peak demands.

- Best-fit decreasing (BFDmax and BFDsum): A heuristic which sorts the list of services by their resource demands in descending order before each placement. BFDsum sorts on the basis of accumulated demands over time while BFDmax sorts on the basis of peak demands. Additionally, the list of servers is sorted in ascending order by their remaining capacity each time a service is allocated.

- Genetic algorithm (GA): A metaheuristic which explores the search space in a number of generations, guided by the solution fitness function. Individuals are encoded

service-centric.

- Grouping genetic algorithm (GGA): A metaheuristic which explores the search space in a number of generations, guided by the solution fitness function. Individuals are encoded server-centric.
- Genetic algorithm with first-fit (GA_FF): A hybrid algorithm where the input sequence to the FFD is optimized using the GA.
- Genetic algorithm with best-fit (GA_BF): A hybrid algorithm where the input sequence to the BFD is optimized using the GA.

The algorithms are implemented in a standard manner according to the pseudo code and explanations given in Section 2.2.4. In the following, the idea behind each algorithm is briefly introduced.

Heuristics

The FFD and the BFD are popular heuristics to solve bin packing problems (Stillwell et al., 2010). In general, both algorithms attempt to allocate services with highest resource demands first. In the case of FFD, the first server on which sufficient capacity remains is chosen for allocation. In contrast, the BFD, chooses the best server, that is, a server on which smallest possible capacity remains. For the sorting of services, multiple dimensions are to be normalized. As stated in Section 2.2.4, the time dimension is normalized by taking the maximum or the sum of the demands across all time intervals. The resource dimensions, on the other hand, are normalized by means of scaling factors which scale all additional resource dimensions according to their ratio to the first resource dimension. Hence, the scaling factor s_1 equals 1 while the remaining scaling factors s_r are calculated case-dependent on the basis of the capacity limits of all servers (cf. Equation 3.4).

$$s_1 = 1, s_r = \frac{\sum_{h \in H} h(1)}{\sum_{h \in H} h(r)}, 1 < r \leq k \tag{3.4}$$

By means of the scaling factors, service demands for additional resource dimensions can be transformed into their equivalents on the basis of the first resource dimension. These values are summed up in order to allow for sorting a list of services by their demands as required by the heuristics before each allocation.

Constraints can be incorporated by means of pre- and postprocessing of the input data. For example, a maximum utilization constraint reduces the server capacity by a defined percentage before running the heuristics on the preprocessed data. Alike strategies to incorporate constraints were tested in a dedicated study and turned out to be inefficient, particularly with respect to quality requirement Q4 (Bosse et al., 2019). New constraint types may arise in the future and the respective processing logic to reach the desired state is to be designed manually for each type. As a result, heuristics are to be preferred for unconstrained problem instantiations. Further conclusions, however, are subject to evaluation.

Metaheuristics

In order to test the suitability of metaheuristics for the SPP, the popular genetic algorithm is used. It represents a well-established algorithm to solve optimization problems (Bäck, 1996; Yusoh and Tang, 2012); its general principle was introduced in Section 2.2.4. A main difference between the genetic algorithm and the grouping genetic algorithm lies in the way individuals are encoded. The following strings exemplify the encoding of a solution candidate which holds five services that were allocated on three servers as used by the GA and the GGA. In this example, services 1 and 4 were placed on server 1, services 3 and 5 were placed on server 2, and service 2 was placed on server 3:

GA: [1,3,2,1,2] GGA: [{1,4},{3,5},{2}]

Encoding used by the GA was presented, e.g., in Stillwell et al. (2010) and Rolia et al. (2003). The GGA encoding was applied, e.g., in Falkenauer and Delchambre (1992) and Xu and Fortes (2010). As shown in the example, the encoding of the GGA seems to depict the problem more naturally as bin packing generally represents a grouping problem (Xu and Fortes, 2010). The mutation operator, in both cases, assigns a service to a different server. Hence, the GA encoding requires to change a single integer value to represent another server while the GGA shifts a random service number to another group of services, i.e., to another sever. For recombination, uniform crossover is applied in both the GA and the GGA. Accordingly, each service will either remain on a server as defined by the first parent or by the second parent; the decision is made randomly. While this procedure generates a consistent child in the GA, the GGA encoding may result in children which show duplicate or missing services as either a group of services from one parent or from the other parent is selected. For this reason, the GGA applies the following three steps as part of recombination:

1. Uniform crossover: Groups of services (i.e., sets of servers) are exchanged between two parent candidates, producing one child.
2. Deletion: In the child, duplicate services are eliminated by deleting their second appearances.
3. Reinsertion: In the child, missing services are inserted on servers with sufficient remaining capacity according to a first-fit heuristic.

Selection is performed on the basis of the fitness function, presented in Section 3.2.2. If constraints limit the solution space, penalty factors or repair mechanisms are well known means to guide the search process into areas of feasible solutions. As pointed out by Coit and Smith (1996), penalization is to be preferred since repair mechanisms would prevent infeasible solutions from being passed into the next generation and, therefore, introduce a bias into the search process. Accordingly, the fitness function was designed to incorporate penalty factors for constraint violations as defined in Section 3.2.3. Furthermore, all penalty factors are multiplied by a generation-dependent temperature factor $\frac{gen}{maxGen}$. This

way, violations are tolerated in the beginning (to exploit a great fraction of the search space) but increasingly penalized when approaching the end of the optimization process (Coit and Smith, 1996).

Contribution to research question 3
Industry experts confirmed placement constraints to be essential for successful resolution of the SPP. Therefore, ten different types of placement constraints were derived from both the scientific and the practical community (cf. Table 3.2). Metaheuristics represent flexible and well-extendable solution algorithms for constrained instantiations of the SPP since they do not require to implement respective ways for achieving constraint compliance manually. Instead, simple checks are performed as part of the fitness evaluation where violations of constraints are penalized depending on the violation degree and the optimization progress. This way, infeasible solutions are less probable to be included into subsequent populations. Furthermore, individual penalty factors allow to weight placement constraints based on the consumers individual preferences.

The GA performs a tournament selection, as described in Section 2.2.4, where t solution candidates are chosen randomly to build a tournament. Those individuals are ranked using the fitness function. From this subset, each individual is selected with a probability p_t proportional to its rank i, as expressed in Equation 3.5.

$$p_t(1 - p_t)^{i-1} \tag{3.5}$$

This selection was chosen to cope with the problem of dominance since the fitness value does not directly influence the selection probability (Miller et al., 1995). As an example, if the selection probability p_t was set to 0.9, the most fittest individual (i=1) is selected with a probability of 90% while the second-fittest individual (i=2) is selected with a probability of 9% and so forth. This process is repeated until the configured population size *pop* was reached. The GGA, in contrast, uses stochastic universal sampling as proposed by Baker (1987) and described in Section 2.2.4.

Hybrid algorithms

Greedy heuristics depend heavily on the ordering of items when packing the bins. As pointed out by Lewis (2009), at least one ordering of items exist which allows a greedy heuristic to identify an optimal allocation. However, this optimal ordering may not be found by sorting the list of items by their demands in a descending order. Instead, a genetic algorithm may be able to identify suitable input sequences to the first- and best-fit heuristics, resulting in two variations of an hybrid algorithm (Liu et al., 2008; Stillwell et al., 2010; Hallawi et al., 2017). Here, a list of services, sorted in a specific order, i.e., a string of size m, represents a solution candidate. Those services are then passed to a first- (GA_FF) or best-fit (GA_BF) heuristic which gains an allocation of services to servers that will be subject to fitness evaluation as described for the GA. For the

recombination operator, edge recombination is used as proposed by Whitley et al. (1989) and further described in Larrañaga et al. (1999). This recombination can be used to recombine permutations as required by the hybrid algorithms which pass a simple list of services to the heuristic. In contrast, the GA and the GGA must recombine full allocations. The mutation operator is implemented by swapping two services, resulting in a change of the ordering. For selection, similar to the GA, tournament selection is applied.

3.2.5 Summary

The problem formulation, which represents the SPP, may be addressed by heuristics or metaheuristics. In order to ensure practical applicability it is essential to allow the representation of placement constraints which may exist in the application environment. Ten types of constraints were identified on the basis of scientific and practical input. While heuristics have been studied to efficiently solve problem instantiations which solely include a single resource constraint, metaheuristics are expected to deal well with problems that are subject to varying types and numbers of constraints. Here, penalization of constraint violations (as part of the fitness evaluation) is generally the preferred way to produce feasible solution candidates as opposed to repair mechanisms. In order to compare and evaluate the solution quality of the most commonly used algorithms, four heuristics, two variations of a genetic algorithm and two hybrid algorithms were selected to solve the SPP.

3.3 Service performance prediction

The algorithms introduced in the preceding section generate solution candidates to the SPP which minimize operations costs c_o. According to functional requirement F2, these candidates must be evaluated with respect to the performance of a business transaction in order to obtain the costs c_{sla} which are associated to SLA violations. This task requires performance models whose construction and validation is carried out by a capacity management provider. Therefore, the structure of this section is formed by the lower swimlane of the process model depicted in Figure 3.2. Since the provider applies domain knowledge when constructing models, a number of expert interviews were conducted along with the data preparation phase in order to interpret given metrics and to design filters. These are referenced at the corresponding steps within the process. A subsequent analysis of future scenarios closes the section.

3.3.1 Data understanding and preparation

As concluded in the ITIL service design publication, “Many SLAs have user response times as one of the targets to be measured, but equally many organizations have great difficulty in supporting this requirement.” (Hunnebeck et al., 2011, p. 170) With respect to this issue, Gmach et al. (2008) states that performance models help to identify resource capacities that are required to satisfy service level objectives under a given extent of service usage.

Expected response times may be estimated before solution deployment and possible SLA violations can be avoided or, in some cases, willfully accepted. Consequently, performance models are assumed to be a useful instrument for evaluating solution candidates with respect to the minimization of operations costs c_o and SLA violation costs c_{sla}. According to the classification given in Section 2.3.2, machine learning-based techniques which utilize shared performance counters were selected as part of functional requirement F8. Using this technique, it is the goal of this section to construct a validated performance model.

With respect to the scope of a model, Cherkasova et al. (2009) points out that predictions need to be made on a transactional basis and can hardly be generalized to the system layer. This holds also true for the cross-environmental approach taken in this work since a considerable share of custom business transactions represent components of a COTS EA which are not comparable across implementations (cf. Section 2.1.1). Therefore, the data basis used for model training must be limited to specific standard transaction types, preferably used by a large number of users. Validated models can be applied to evaluate solution candidates with respect to particular business transaction performance. In order to generate input data for model application, the historical workload of running services must be linked to the newly allocated servers. Prediction output represents the resulting performance effect and, therefore, can be used to compute the expected number and degree of SLA violations for each solution candidate. For example, different maximum server utilization levels may have been configured as placement constraints when solving the SPP; those result in a set of alternative solution candidates whose total cost analysis is enabled by the predicted information.

As the application area of the resulting performance model is generally limited by the variety of used training data, a broad and heterogeneous spectrum of environments ensures a sufficient training data basis. As argued in Section 2.1.4, today's monitoring tools provide useful insights into transaction activity across different components in multitier systems (Cherkasova et al., 2009). Hence, workload traces hold great potential to bridge the gap between software performance engineering and capacity management (Rolia et al., 2005). As a consequence, the amount of available data sources is increasingly shifting the focus from the model selection phase to the data preparation phase. Russell and Norvig therefore conclude that "it makes more sense to worry about the data and be less picky about what algorithm to apply." (Russell and Norvig, 2010, p. 27).

The training data, in the present work, is formed by hourly performance records for a business transaction. Performance of EAs can be expressed by means of throughput or response time metrics. While batch applications are throughput-sensitive, dialog applications are rather response time-sensitive (Goudarzi et al., 2012). As SLAs typically refer to services with user interactions (Gmach et al., 2008), response times are assumed to be most most relevant for the end user experience. Therefore, the outcome y (target) represents the hourly mean response time per dialog step where a dialog step represents any kind of user activity within a business transaction. The target is a function of a set

of n hours (observations), each having m attributes a which characterize the observation:

$$y = f(a_1, a_2, \ldots, a_m) \tag{3.6}$$

The attributes must integrate metrics from all layers involved in transaction execution. Pinheiro et al. (2001) points out that execution time is hard to predict and depends heavily on application characteristics. Despite the application server, transaction latency comprises CPU and queue times from the database server (Cherkasova et al., 2009). While current approaches rarely take both hardware and application characteristics into account, (Matsunaga and Fortes, 2010) argues for a large attributes space which should take detailed application-specific characteristics into account while also considering the heterogeneity of systems. After testing and comparing several ML algorithms to predict the execution time of applications, the authors generally advice to include as many attributes as available in order to enable the algorithm to decide on their individual importance. Hence, monitoring data from three different performance dimensions, often considered separately, were joined in order to create the attribute space.

- **Workload characteristics** refer to transaction usage indicators and associated resource demands. Metrics include data traffic sent between the client, the application, and the database server in order to serve transaction execution. Furthermore, user activity and dialog steps are covered.

- **Resource capacity** refers to the CPU and memory capacity of involved servers. Furthermore, the year of data collection may represent basic server generation classification.

- **Meta data** refers to additional information which describes the measurement itself, the type of involved systems and the topology. Therefore, this dimension also refers to the mapping of physical to logical components, e.g., which application instance is running on which server.

The selection process of attributes from the three dimensions was guided by three performance experts of the target domain in multiple focus group discussions. Table 3.3 lists the consolidated result across all dimensions which represents the feature space for model training. While some of the attributes originate from direct measures of the EA, others result from an automated preprocessing of these measures as described in Wilhelm (2001) and Wilhelm (2003). However, unfiltered use of these attributes may lead to the risk of inaccurate models as outliers are often present in productive monitoring data. In that case, performance-related decisions in capacity management would be made on the basis of erroneous data and possibly provoke serious damage. In this regard, data-cleansing procedures help to exclude records which correspond to abnormal application behavior (Cherkasova et al., 2009).

An example for atypical observations are time intervals in which system management activities interfere normal operations. Cherkasova et al. compare the accuracy of a

Attribute	Short description
	Target value
RESPTI	Mean response time per dialog step in ms.
	Workload characteristics
DSCNT	Number of dialog steps
DBSU	Number of database service units (select, insert, update, delete).
DBKB	Data written and read by the DBMS in KB.
CLBY	Data written and read by the client in bytes.
OTHERKB	Data written or read by other Services.
DBSU_OTHER	Number of database service units executed by other services.
DSCNT_OTHER	Number of dialog steps executed by other services.
TACOUNT	Frequency of transaction execution.
USERLOW	Number of low activity users.
USERMEDIUM	Number of medium activity users.
USERHIGH	Number of high activity users.
INST_MEM	Memory demand in mb.
APP_SAPS	CPU demand of the application service in SAPS.
DB_SAPS	CPU demand of the database service in SAPS.
	Resource capacity
APP_MAXSAPS	CPU capacity of the application server in SAPS.
APP_MEM	Memory capacity of the application server in MB.
APP_CPUNUM	Number of CPUs in the application server.
DB_MAXSAPS	CPU capacity of the database server in SAPS.
DB_MEM	Memory capacity of the database server in MB
DB_CPUNUM	Number of CPUs in the database server.
YEAR(TS)	Year of the measurement.
	Metadata
MEAS_DAYS	Number of days the environment was monitored.
TYPE	EA type (can be PROD, QA, or DEV).
INST_COUNT	Number of services, running on the application server.
APP_VMINFO	Virtualization details of the application server.
DB_VMINFO	Virtualization details of the database server.
TS	Time stamp of the observation.

Table 3.3: Attribute space.

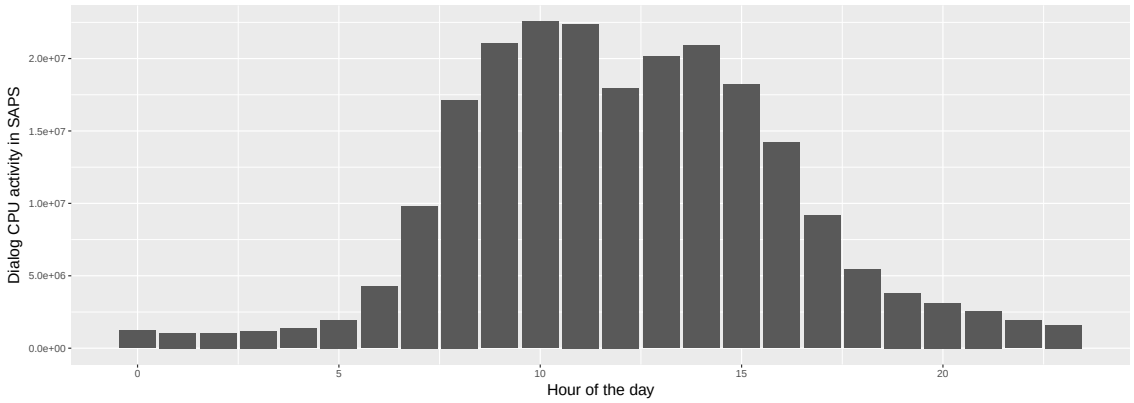

Figure 3.3: Distribution of CPU acticity for dialog tasks across the day.

model that predicts CPU demands and observed a doubled error if time frames of system management were included into the training data basis. Such activities are usually scheduled orthogonal to transaction processing activities, i.e., on the weekend or at nighttime. Hence, observations were first filtered to include time periods of typical dialog activity only. To obtain hours of regular dialog activity, an analysis of the task type distribution was performed across all measurement points. The bar chart in Figure 3.3 illustrates the result according to which least dialog activity occurs at night between 11pm and 4am. Therefore, as an initial data preparation task, data was limited to observations made on weekdays between 5am and 10pm. Next, data is to be grouped by business transactions in order to build performance models according to functional requirement F8. Of course, business transactions of interest depend on the individual SLAs given by the capacity management consumer. However, model accuracy is influenced by the amount of training data and, therefore, by the frequency of transaction usage. In order to exemplify the model creation and evaluation process, a business transaction was chosen whose performance is business-critical on the one hand. On the other hand, sufficient training data must be ensured. Therefore, consulted domain experts limited the space of transactions to the SAP module which covers *sales and distribution* processes as most customers would consider those processes to be performance-critical. Within this module, a data-driven approach was taken in order to identify the transaction type of most frequent use. Figure 3.4 reveals the business transaction type *Change sales order* (SAP transaction code VA02) to be the most frequently executed transaction within the considered module. Across all executions of VA02, different complexity classes can be distinguished on the basis of the number of database service units (Wilhelm, 2001). The classification of transactions regarding their complexity is also described, e.g., by Menascé et al. (2004) for model-based performance prediction. As confirmed by the preliminary work of Wirth (2015), prediction accuracy increases significantly if models are limited a priori to a particular complexity class. In accordance with the author's results, a class of medium complexity (3 out of 5) was chosen to proceed with. The target value across all observations associated to VA02 of complexity class 3 ranges between 36ms and 60,078ms and the average response time accounts to 598.95ms. In general, the classification of observations into groups of desired and un-

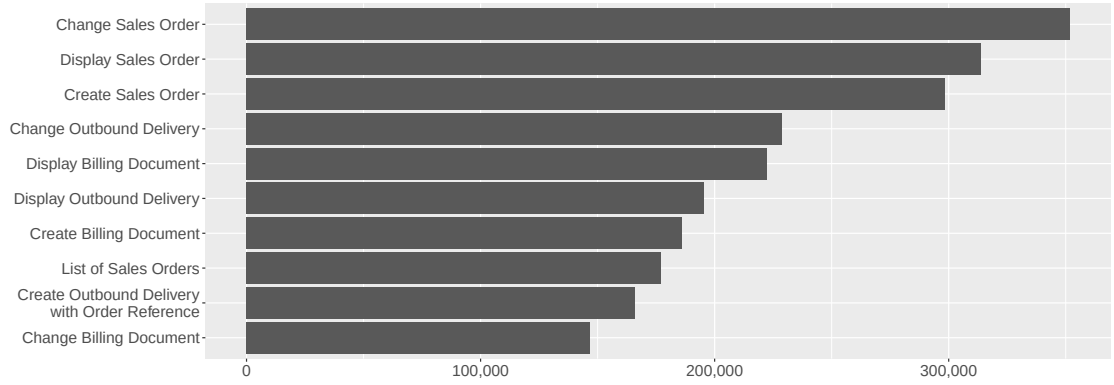

Figure 3.4: Frequency of transaction usage across the sales and distribution module.

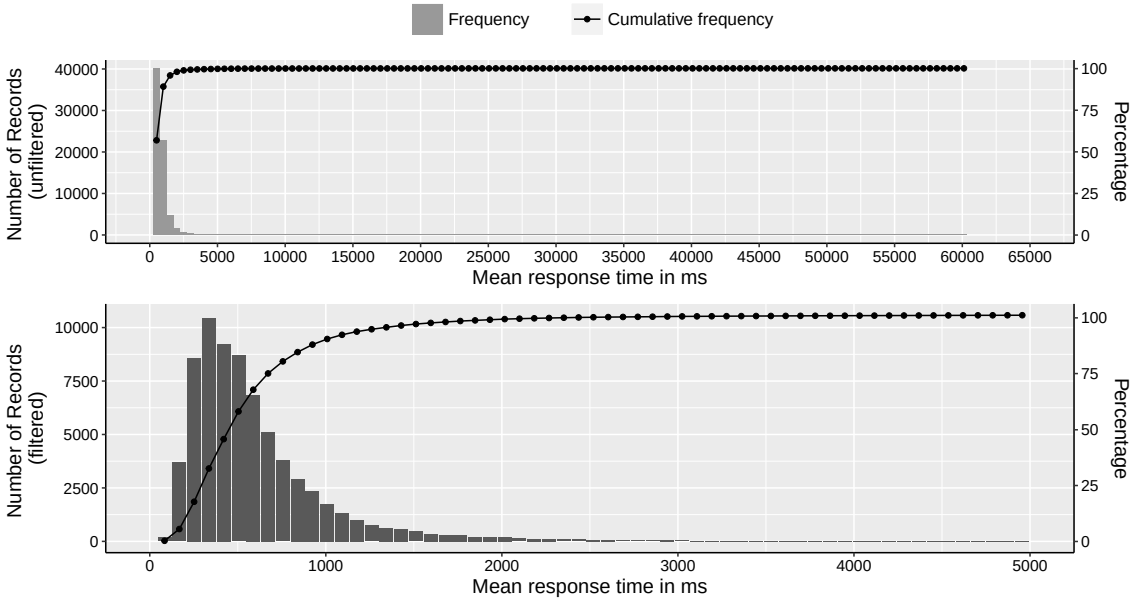

Figure 3.5: Histogram of the target value before and after outlier removal.

desired outliers can hardly be automated. Therefore, histograms were built to show the distribution of response times (cf. Figure 3.5). On this basis, domain experts were consulted in order to find a threshold which splits data into two classes. As a result, response times above 5,000ms were classified as undesired outliers. The resulting distribution of the target value is shown by the lower histogram in Figure 3.5. Likewise, additional filters were applied as part of the data preparation phase with the objective to keep as much observations as possible. Table 3.4 summarizes the resulting value ranges for each attribute that has been filtered along with a short description of the filtering intention. The last column indicates the percentage of data that was excluded by the respective filter.

Besides the filters listed in Table 3.4, additional data quality filters were used to exclude erroneous data, i.e., servers with 0 capacity in one of the resource dimensions, possibly resulting from corrupt configurations. As a result, data is prepared to represent valid observations only. The final data set comprises 21 features and 1 target variable. The

Attribute	Range	Intention of filtering	Perc.
RESPTI	<5000	Cut response time outliers	0.31
APP_SAPS	>80	Ensure min. CPU activity on application server	10.13
DB_SAPS	>80	Ensure min. CPU activity on database server	10.52
DSCNT	>10	Ensure min. dialog activity	26.61
MEAS_DAYS	>2	Exclude test measurements	1.30
TYPE	PROD	Limit to productively running EAs	3.03
APP_VMINFO	NULL	Exclude virtual machines	6.58
DB_VMINFO	NULL	Exclude virtual machines	3.99
WEEKDAY(TS)	0-4	Exclude maintenance time frames (Mo. to Fr.)	2.67
HOUR(TS)	5-22	Exclude maintenance time frames (nighttime)	4.08
USERS	>2	Ensure min. user activity	1.67

Table 3.4: Attribute filters.

mean target value accounts to 571.87ms with a standard deviation of 465.28. In addition to filtering, some models require normalized data, i.e., numeric features which are scaled to be on the same magnitude. For example, Support vector machines are known to be sensitive to the scale of input feature while Regression trees and Random forests do not require normalization (Müller et al., 2016a, p. 98ff.). Likewise, feature selection may be inherent to model training (e.g., in the case of Random forests) or, alternatively, carried out in advance. Therefore, techniques for normalization and feature selection are applied if appropriate as mentioned in the following subsection.

3.3.2 Model training and evaluation

The goal of this task is to construct a model which fulfills the domain-specific accuracy requirements. Here, model training refers to the learning phase which ML algorithms pass through in order to learn a concept from historical data and transfer the dependencies into a model (Matsunaga and Fortes, 2010). Therefore, a suitable prediction model type must be selected from an individual range of considered models (Bishop, 2006). Many solutions were published where ML was applied to different scenarios of application performance prediction (Gupta et al., 2008; Duan et al., 2009; Huang et al., 2010; Matsunaga and Fortes, 2010; Niehorster et al., 2011; Venkataraman et al., 2016). However, application types, attributes and target metrics widely vary and hinder to compare evaluation results with respect to a general recommendation of a suitable model type (Matsunaga and Fortes, 2010). As stated by Niehorster et al. (2011), complex applications often show a non-linear behavior which limits the space of applicable models to more complex techniques, e.g., those which perform recursive partitioning before the actual regression (cf. Section 2.3.3). Consequently, both Matsunaga and Fortes (2010) and Niehorster et al. (2011) obtain good results in the targeted domain with SVMs. An empirical comparison of supervised learning algorithms was published by Caruana and Niculescu-Mizil (2006). Here, remarkable results across different data sets were obtained by Random forests as an example of a bagging algorithm. However, a boosting technique (Boosted trees) is

also among the top algorithms. In contrast, worst results were obtained, among others, by decision trees. The work performed by Wirth (2015) confirms these findings for the domain of SAP EAs. Simple linear regression and regression trees were not able to achieve suitable levels of accuracy (MAPE >30%). Random forests, meanwhile, reached a MAPE around 25%. To identify suitable model types for predicting response times in the domain of a COTS EA, different ML models were trained and validated in a preliminary work, using real monitoring data of a structure similar to the one described in the preceding section. Results were published in (Müller et al., 2017c) and revealed lowest errors when applying a Random forest.

Thus, based on the mentioned preliminary work, Random forests and Support vector machines were chosen to build performance models. Furthermore, following the results of Caruana and Niculescu-Mizil (2006), Regression trees that are boosted using the ML meta-algorithm AdaBoost are tested too. For every model type, 5-fold cross validation with 20% test data was applied as also done by Caruana and Niculescu-Mizil. Respective models and their function were introduced in Section 2.3.3. In the following, error metrics and respective hyperparameter configurations are briefly introduced for each model type before an overall comparison of model results is provided.

Random forests

Random forests do not require to normalize input features and feature selection is performed implicitly at each split as introduced in Section 2.3.3. Therefore, in the initial run, neither feature selection nor normalization was applied. The number of trees to learn (parameter *ntree*) depends on the data volume. In general, large numbers produce more stable models but require more memory while small values lead to the risk that single input records are used rarely for predictions (Liaw and Wiener, 2018). Hence, for small data sets a number of trees around 50 is proposed while large data sets may require to grow more than 500 trees (MGET, 2019). Wirth (2015) addressed a similar business problem and observed a decreasing error with an increasing number of trees up to a threshold of around 200 trees. However, the volume of data used by Wirth is about 30% smaller than the one used in this thesis. Furthermore, memory and computing time is not a great concern in the given context of offline optimization. Therefore, Random forests were trained with a number of 500 trees in favor of accuracy. At each split, seven variables were randomly sampled to be considered for the splitting. This corresponds to the default value of the parameter *mtry* for regression trees which is defined as the number of variables divided by 3 (Liaw and Wiener, 2018). The minimum size of a terminal node (parameter *nodesize*) defaults to 5 and was left unchanged. Model validation resulted in an averaged MAPE of 26.08% across the 5 folds. The RMSE accounts to 303.88 and R^2 equals 0.57. In subsequent test runs, neither feature selection nor normalization efforts improved the accuracy since Random forests generally require very little tuning (Friedman et al., 2017).

Support vector machines

In SVMs, the kernel function maps the input space into a linear separable feature space as introduced in Section 2.3.3. The selection of an appropriate kernel function depends on the input space and is generally carried out by means of experiments (Hofmann, 2006). In many cases, however, accuracy of models using a linear kernel falls below Radial basis function (RBF) kernels (Keerthi and Lin, 2003). For this reason, a Polynomial kernel (PK) and an RBF kernel are tested. The PK, with default settings (cost = 8, gamma = 1/number of attributes, degree = 3), resulted in a MAPE of 39.30%. Hyperparameter optimization (HPO) efforts with costs varying over the string [2, 4, 8, 16, 32, 64, 128, 512, 1024] and gamma varying over the string [0.1, 0.5, 1, 2, 2.5, 3, 3.5, 5, 8, 10] did not improve the error so that further attempts were not carried forward, given the clear superiority of the Random forests' error metrics. For the RBF, best results were achieved using Z-score normalization and without feature selection, resulting in a MAPE of 25.90%. Similar errors were achieved when additionally selecting features on the basis of a Regression tree (method *Rpart1SE*) and using Recursive feature elimination (RFE) as provided by the R package *Caret* while keeping the Z-score normalization. Appropriate values for the hyperparameters costs and gamma were explored in a sensitivity analysis where best accuracy was reached by costs = 1.5 and gamma = 0.5. However, the optimization did not improve the error significantly.

Boosted trees

The ML meta-algorithm AdaBoost was used in conjunction with 200 regression trees as weak learners. Details of the algorithm implementation are described in Drucker (1997). In order to avoid over-fitting, the minimum number of samples required to perform additional splits was raised from its default 2 to 4 and the maximum depth of a tree was limited to 16. Without any feature selection or normalization, AdaBoost could improve results achieved by the bagging technique of the Random forests significantly. While the RMSE goes slightly down from 303.88 to 294.72, the MAPE accounts to 19.57%. Accordingly, R^2 improved to 0.6.

Model selection

At first glance, all model types achieved suitable errors as the MAPE falls below the threshold of 30% defined in quality requirement Q2. Nevertheless, error metrics of the trained models are to be compared in order to select an appropriate model for release. One of the simplest forms of visual evaluation is to ablate measured and predicted values in a two-dimensional space as illustrated in Figure 3.6. Here, results of the best fold for each model type are used. An optimal model would indicate a straight line through the origin. In this sense, closest to optimum are the Boosted trees while RF and SVM with RBF kernel show similar results. In contrast, the SVM with PK predict negative response times in some cases and, therefore, fail to minimize the distance to the original line. As

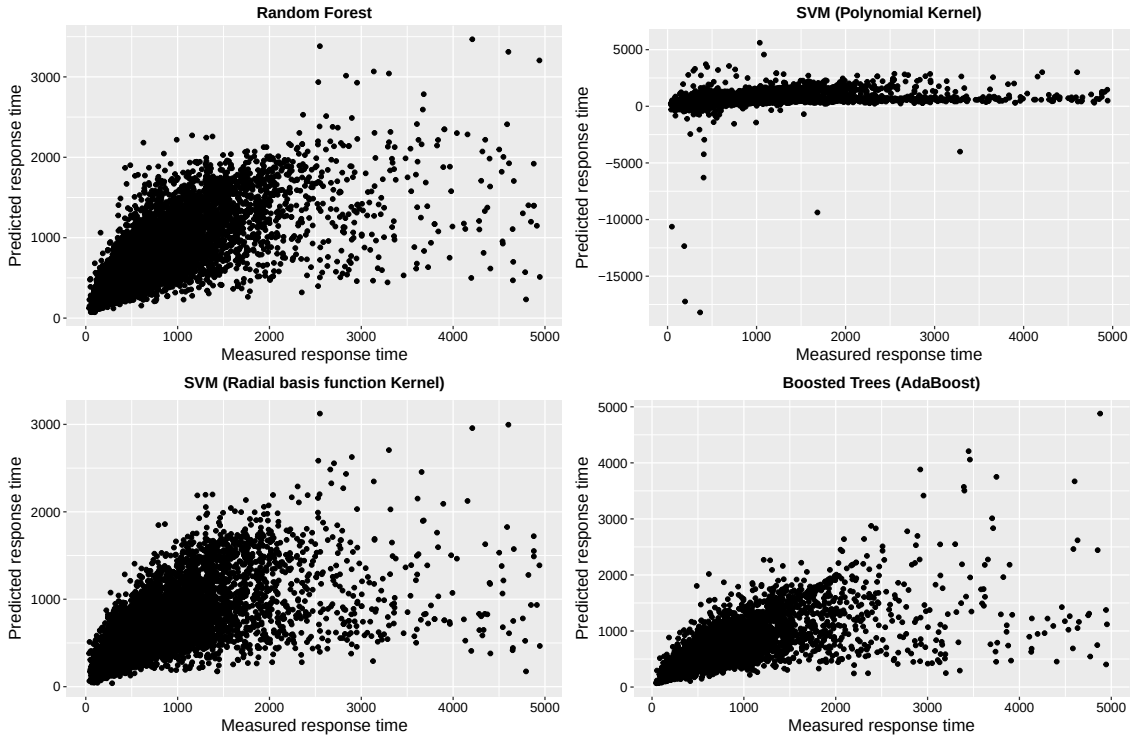

Figure 3.6: Fitted line of measured and predicted values.

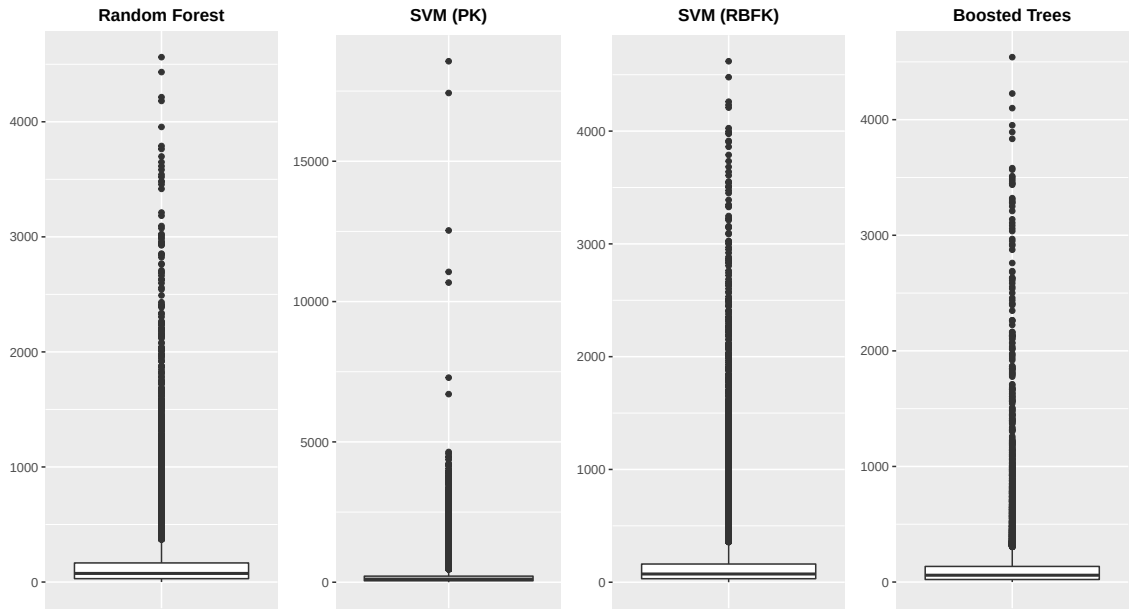

Figure 3.7: Distribution of absolute errors.

another way to visualize accuracy, box plots of the absolute errors may help to illustrate their distribution. Figure 3.7 shows that predictions (except for the SVM with PK) range from 0 ms to 4561.76 ms where the maximum error was produced by the Random forest. SVM with PK predict with errors up to 18,563.05 ms which is surely not acceptable with respect to the given domain and the average response time of 571.87 ms. The absolute errors of the Random forest and the SVM with RBF kernel show median values between 72.9 ms and 74.8 ms while their interquartiles (50% of the data) range between 29.4 ms and 166 ms. Here, Boosting trees stand out with a median error of 59 ms and an interquartile range between 22 ms and 135 ms.

In 2003, Bi and Bennett proposed the Regression error characteristics curve (REC) on the basis of the Receiver operating characteristic (ROC) curves known from the field of classification. The instrument is intended to compare regression errors of different models and was also used by Matsunaga and Fortes (2010). REC curves plot a threshold of error tolerance on the x-axis against the percentage of records that were predicted with errors below this threshold. Figure 3.8 shows the REC curve for the tested four model types. To interpret the REC curve, first, an error tolerance level is to be defined on the x-axis. Next, the percentage of records that was predicted within this level can be taken from the y-axis for each type of model. Therefore, in almost 100% of the cases the error is below 100%. In a small number of cases, however, absolute percent errors accounted to more than 100%. According to quality requirement Q4, the mean absolute percent error must fall below 30% in order to be accepted. The REC curve additionally shows that, in almost 80% of the cases, the error did not exceeded this threshold when using Boosted trees. Likewise, Random forests undercut this value in around 70% of the cases while SVM with PK show worst error characteristics with around half of the cases exceeding the tolerance level of 30%. As an additional analysis, density plots indicate if predictions show a similar distribution when compared to the measured values. Density lines are given in Appendix A for each model. Finally, information on feature importance was computed in order to provide insights regarding the influence of particular workload or hardware characteristics on the execution time. Results are also meaningful for practitioners who work in the field of capacity planning, e.g., when sizing servers according to given service level objectives. In the present case, feature importance levels were shared within a group of performance engineers who confirmed the practical utility of the results which could be applied in current consulting projects, e.g., as part of bottleneck analyses. Many regression techniques allow to calculate variable importance during model training. Here, percentages for each feature, representing its influence on the result, are provided. A model-independent procedure to compute variable importance is to exchange the value of a feature by a random value within the range found in the training data. A high distance between the two regression results (original value and replacement) indicates a strong influence of the modified feature. However, various model-dependent ways exist to determine feature importance. For example, in regression trees, features which were chosen to split the tree early, have stronger influence on the prediction result while lower

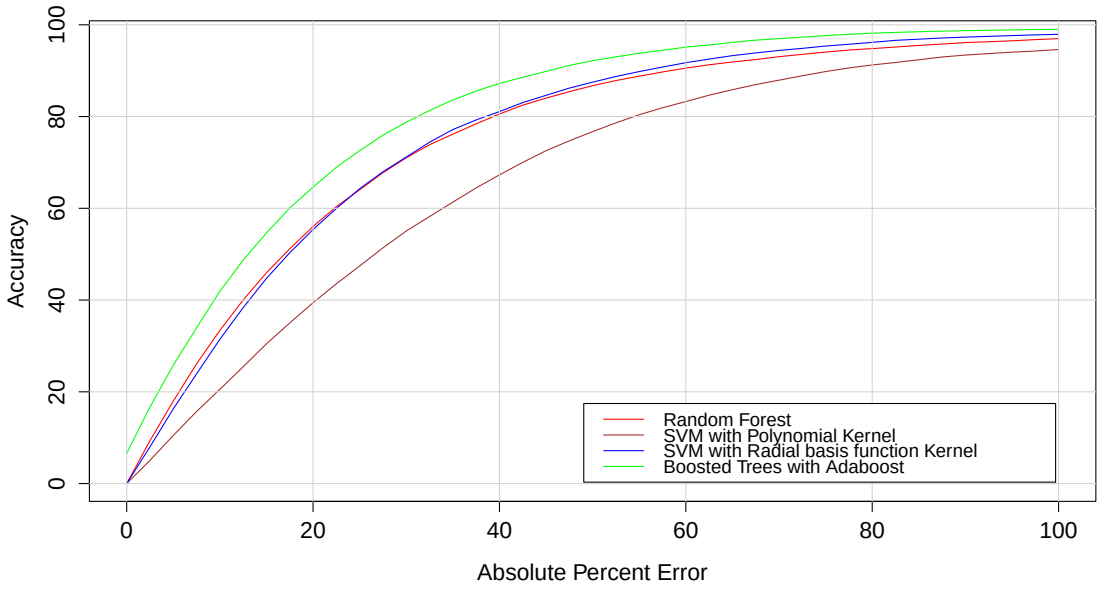

Figure 3.8: Regression error characteristics curve.

features only affect the respective branch. Details on the exact computation method are out of scope in this work, however, results across a variety of feature selection techniques are given by the box plots in Figure A.4 of Appendix A. To summarize the results, no feature was considered to be unimportant in every case. The high variance of the box plots shows that different feature selection techniques vary greatly in their results. However, a high median and small interquartile range are good indicators for an important feature, thus, tendencies are recognizable. Examples are the number of CPUs of the database and application server. Most important feature represents the volume of data written and send to the database (DBKB) with a median importance of 100% and a mean importance of 65.64%. This feature was also rated by the AdaBoost feature importance function to be among the top three features. Here, most important features are the number of dialog steps, followed by the number of database service units (cf. Figure A.3). Such meta information may be provided by the capacity management provider as additional reference material.

To conclude, Random forests and SVMs perform similarly well although SVMs require some additional efforts to identify a suitable kernel function and to optimize hyperparameters. However, for the tested models using the given data set, HPO did not improve the result significantly. In this regard, Random forests and Boosted trees produce acceptable results out of the box when growing a sufficient number of trees. Across all tests, Boosted trees using AdaBoost achieved best error metrics with a MAPE that is 10 percent points under the defined level of tolerance (quality requirement Q4). Therefore, based on the findings, Random forests and Boosted trees are applicable model types where the former allows to train trees in parallel and the latter provides best accuracy. The best achieved

RMSE accounts to 294.72. When comparing the RMSE to the standard deviation of the measured target value (465.28) it becomes clear that mean distances between predictions and measured values are lower than distances between measured values and their average. Therefore, modeling effort turned out to be useful in comparison to a rather naive estimation on the basis of the average value.

Finally, in order to analyze the generality of the results, the two model types AdaBoost and Random forests were used to train and validate performance models for the most used transactions of medium complexity as listed in Figure 3.4. The resulting REC curves are illustrated in Figure A.2 of Appendix A. Mean absolute percent errors are provided in Table A.1. In the case of AdaBoost, all tested models show a MAPE below 30%.

Contribution to research question 5
Non-linear ensemble models which apply bagging or boosting techniques are suitable instruments to predict the response times of a business transaction. In the tested example of the frequently used SAP transaction to change sales orders, mean response times could be predicted with a mean absolute percent error of 26% when using Random forests and 19% when applying Boosted trees in conjunction with AdaBoost. Feature space was formed by application-specific workload characteristics, resource capacity limits, and meta data which explains the data origin.

The results are consistent with the work of Drucker (1997) who compare boosting and bagging techniques utilizing Regression trees. The authors state that boosting revealed either equal or lower errors than bagging across all tests. In general, ensemble methods which combine a number of trees perform well without much tuning (Friedman et al., 2017; Carr, 2005). Therefore, means to improve the accuracy lie mainly in the data acquisition and preparation phase. Here, a trade-off between the number of features and the data volume was faced. While, for some observations, additional features would have been available, the total amount of observations decreases when considering those features for model training. For example, a detailed composition of consumed main memory is available for a particular type of services only. Therefore, it is subject to future research activities to test models that were trained on a smaller subset of data comprising a higher number of available attributes. Additional research directions with respect to the prediction of performance are given in Section 5.3.

3.3.3 Model release and application

As defined in the capacity management process (cf. Figure 3.2), if the accuracy of a tested model is acceptable, it is released by the capacity management provider. Breiman and Cutler (2019) point out that models do not produce single truth but a “computer generated guess” which support in comprehending a given problem. Therefore, the model is to be stored in the APM-KB along with accuracy metrics which enable interpretation of results obtained when applying the model. Moreover, the capacity plan which accumulates the process output, according to ITIL, must contain information on used methods for any

forecasts in order to assess the results' credibility (Hunnebeck et al., 2011).

Released models are applied by the capacity management consumer in order to predict the execution times of particular business transactions for a number of alternative solution candidates, each representing a design of services-to-server allocations. This way, solution candidates can be evaluated with respect to their expected fulfillment level of performance-related SLAs. In order to generate the model input data, hourly transactional workload metrics of each running service are linked to the capacity limits associated with the server it was placed on. The resulting records represent a fictional allocation whose mean hourly performance is to be estimated. Therefore, the number of predictions being made equals the number of hours in which the transaction under study was executed by any service. However, as the degree of SLA violations depend on the workload extent, alternative load scenarios may be considered in addition to the scenario which is based on the measured historical workload. Hence, measured workload metrics may be scaled according to a load factor which represents alternative business scenarios.

With respect to the level of detail, i.e., the application area of the model, it must be stated that each model can only be applied to predict execution times of the respective transaction type it was trained with. If multiple business transactions are to be investigated, individual models are to be used. In this case, predictions for the n most critical transactions could be made before summing up the resulting SLA violation costs on the basis of all predicted values. At last, total costs are to be minimized so that the generalization of transaction-specific performance to an overall service performance metric is not mandatory. Model construction on upper layers, e.g., on the level of an instance or a system (cf. Figure 2.1), however, was additionally tested using the two most promising model types Random forests and Boosted trees. Here, error metrics, according to the quality requirement Q4, were not acceptable (MAPE >50%) so that the designed approach, based on the tested data, is not applicable to estimate overall system performance directly. Instead, Mi et al. (2008) proposes to use a mean value analysis (MVA) across different predictions of the individual transaction mix, if harmonization of results is desired. Alternatively, for the model-based prediction of performance metrics on the level of complete instances or systems, white-box approaches as listed in Table 2.2 may be taken.

To summarize, models are applied to predict transactional response times for each load scenario within each solution candidate. On this basis, SLA violation costs can be estimated for each future scenario. A subsequent *Future scenario analysis* aims to identify most cost-effective solution candidates.

3.3.4 Future scenario analysis

Capacity management involves to evaluate alternatives. According to the Intelligent-design-choice decision model formulated by Barnard and Simon (1947), three fundamental phases are required to support decision making:

- Intelligence phase: understand the problem.

- Design phase: generate and evaluate alternatives.
- Choice phase: compare alternatives and decide.

As the heuristics and metaheuristics, designed in Section 3.2, converge to a single solution (Adamuthe et al., 2013), the SPP may be instantiated with different optimization degrees, producing alternative solution candidates as part of the design phase. The step described in this subsection aims to support the choice phase where the effects of decision alternatives should be estimated with a suitable accuracy. Here, the use of previously constructed prediction models allows to map decision variables to objective values. The alternative optimization degrees can be defined by varying target utilization levels as also done by Gmach et al. (2008). For this purpose, the maximum utilization constraint type (cf. Section 3.2.3) provides suitable controls. An additional parameter, as stated in the preceding section, is expressed by different load factors in order to cover alternative business scenarios which may be conceivable for the period of time the capacity plan refers to. Hence, a load scenario which is linked to a solution candidate indicates a possible future scenario, as exemplified in Table 3.5. Each future scenario generates costs that are to be computed as part of the future scenario analysis.

The overall objective is to identify solution candidates which minimize the total costs for a given load scenario. In order to consider the full solution space, load scenarios may be associated with probabilities. Total costs, as argued in the problem formulation of the SPP (cf. Section 3.2.1), comprise operations costs c_o and penalty costs c_{sla} which result from SLA violations. In this regard, Cao and Dong (2014) describe an energy-performance tradeoff since service providers seek to reduce energy costs as part of the operations costs while keeping SLA violations on a low level. The prediction component of this work addresses performance-related SLAs only; other categories such as availability or security can be addressed by modeling respective constraints as described in Section 3.2.3. According to the energy-performance tradeoff, aggressive optimization reduces operations costs but increases the risk of peak loads which cannot be served in accordance to defined service levels. The effect on respective cost types leads to the dilemma delineated in Figure 3.9, which, due to its characteristic shape, is termed Bow tie dilemma of total costs. The consolidation pressure can be adjusted over C_{util}. As each solution candidate imposes a different C_{util}, it is the goal to find a solution among the candidate set s which minimizes

Solution Candidate	**Load Scenario**	**Future Scenario**
A (e.g., high utilization levels)	1 (e.g., historical load)	1
	2 (e.g., doubled load)	2
B (e.g., mean utilization levels)	1 (e.g., historical load)	3
	2 (e.g., doubled load)	4
C (e.g., low utilization levels)	1 (e.g., historical load)	5
	2 (e.g., doubled load)	6

Table 3.5: Solution candidates and load scenarios form future scenarios.

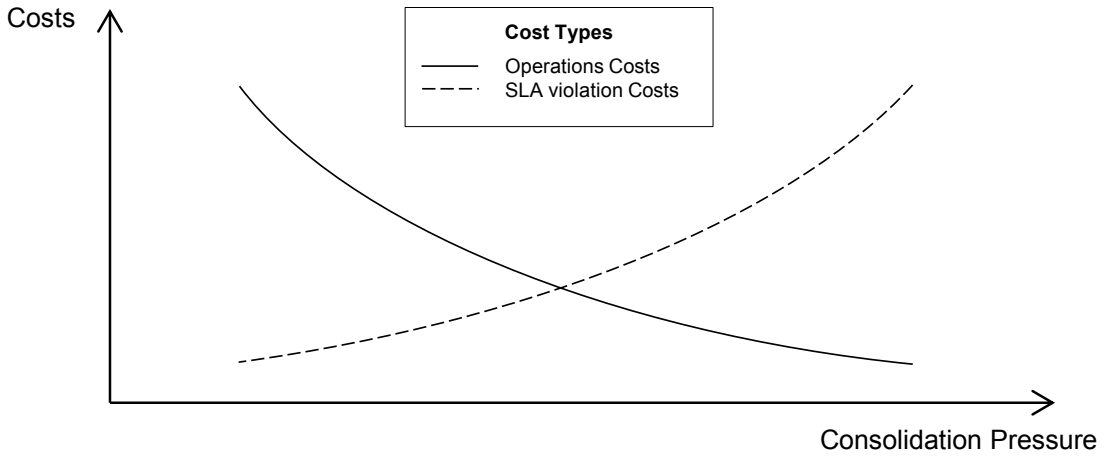

Figure 3.9: Bow tie dilemma of total costs resulting from the energy-performance-tradeoff.

the total costs c_{total}. According to Equation 3.7, the total costs are computed by the sum of operations costs across all n servers h and the SLA penalties.

$$c_{total} = \sum_{j=1}^{h_n} c_{o_j} + c_{sla} \tag{3.7}$$

In the exemplified cost behavior, depicted in Figure 3.9, the point of minimized total costs is represented by the intercept of the two cost lines. Therefore, as pointed out by Hyser et al., a control mechanism may decide to "violate a service level agreement if the financial penalty for doing so was less than the cost of the power required to meet the agreement" (Hyser et al., 2007, p. 8). While SLA penalties are typically derived from monetary metrics, operations costs, as defined in Section 3.2.1, are represented by the required server capacity in SAPS. To enable accumulation and minimization, both cost types are to be normalized on a regular basis such as monthly costs in Dollar. With respect to the operations costs, server capacity limits (e.g., in SAPS) may be translated to monetary costs (e.g., in Dollar) according to a mapping function. The function can be either continuous or discrete where the latter may utilize server classes (e.g., S, M, and L) that are associated with average costs as proposed by Janssen and Marquard (2007). Alike T-shirt sizes are also used in the sizing tool SAP Quick sizer (Loibl, 2015). Server capacity limits for all required servers are outputted as part of a solution candidate to the SPP. Furthermore, operations costs may include a share of additional cost types such as personnel costs in order to represent maintenance effort (Bobroff et al., 2007).

When it comes to SLA violations, Appleby et al. (2001) distinguish between requirements and goals where goal violations do not trigger penalties. Hence, the following refers to requirement violations. In general, SLAs are required to be expressed in a measurable way, which makes the quantification of penalties rather simple. Gmach et al. (2008) point out that fixed step-percentile SLAs are increasingly adapted in the domain of EAs.

Here, a certain fraction of requests is required to show a defined response time where the non-conforming requests entail penalties (Boloor et al., 2010a). This principle is also used in the sample SLA for the average response time of a service in the ITIL Service design publication (Hunnebeck et al., 2011). Hence, on the basis of the predicted response times, requests can be classified into a class of violated SLAs and a class of fulfilled SLAs. Subsequently, the fulfillment percentage and the violation percentage are computable. Percentile-based SLAs require $r\%$ of requests to be processed within a defined time frame, e.g., within x milliseconds. A penalty p is invoked for every m percentage points under fulfillment. Thus, SLA penalties are computed according to Equation 3.8 with v being the number of violated requests, n being the total number of requests, and p being the the defined penalty for each percent point under fulfillment. Requests are defined as periods of execution for a particular business transaction.

$$c_{sla} = \left\lfloor \frac{v * 100}{n} \right\rfloor * p \tag{3.8}$$

In some cases, an additional deadline constraint d_2 is defined which refers to a threshold of response times whose exceedance is generally unacceptable. In this case, if any request exceeds d_2, the complete design is not considerable.

With the computed cost types, Equation 3.7 yields different total costs for each solution candidate which are to be minimized. If probabilities for the alternative load scenarios are known, these can be used in an a priori optimization approach. By means of the input, weighted load scenarios can be aggregated to a story line, representing the most likely series of future events. On this basis, a single solution candidate can be recommended as the cost-optimal design for a story line. If probabilities are not available, unweighted cost estimations provide valuable input to the decision maker when analyzing the future scenarios.

3.3.5 Capacity plan

The results of the capacity management process, in accordance with ITIL, are documented in the capacity plan which represents the output of the designed process. It holds scenarios and forecasts regarding the capacity demand from a business perspective as well as alternative actions to satisfy the demands where alternatives must be associated with costs (Hunnebeck et al., 2011). Accordingly, the capacity plan comprises the following elements:

- **Management summary:** Recommended actions such as the deployment of a particular solution candidate are to be given as part of a short executive summary. Furthermore, total costs for each solution candidate are to be listed in order to make recommendations comprehensible.

- **Future scenarios:** More details on solution alternatives and a cost breakdown are to be provided. As the capacity plan must reflect the business strategy and potential

growth, all forecasts must be denoted and any assumptions made must be clearly mentioned. If, e.g., a scenario of doubled load, as given in Table 3.5, was assumed, it is to be reasoned which component of the business strategy induced this choice. If applied, different probabilities for the alternative load scenarios are to be provided as well. Finally, both SLA costs and operations costs are to be listed for each future scenario.

- **Summary of services and resources:** This section provides information relevant to decision making in more detail. Workload profiles for each service with respect to the considered resource dimensions must be provided. Furthermore, server capacity values are to be listed. For each server, target utilization levels which result from each solution candidate are to be given in the form of peak and average values for each hour of the day. An allocation table summarizes which service would run on which server for each design. Finally, modeled constraints are to be documented.

- **Application area:** The application area targeted by the capacity plan must be indicated, e.g., a specific data center of the organization or a sub-problem (zone) within a specific data center. The latter may apply if the component exclusion constraint (cf. Section 3.2.3) was utilized.

- **Used methods:** To interpret the information, it is necessary to mention how it was obtained. Therefore, forecasts should be documented along with accuracy details of the applied models.

The capacity plan is to be created by the capacity management consumer, using information provided by the provider. The major share of information can be extracted from the APM-KB in an automated way. An exception to this rule are sensitive information regarding the business strategy which is not stored in the APM-KB. The capacity plan may be provided online as an interactive web page or in the form of a report file to decision makers.

3.3.6 Summary

Information on hardware resources and workload characteristics, monitored at a variety of implementations of a COTS EA, form an attribute space that is made up by *shared performance counters*. It could be shown that alike data can be prepared and used to train machine learning techniques for the prediction of response times with acceptable errors. Most promising techniques for the investigated domain are non-linear ensemble techniques that utilize the concepts of boosting (Boosted trees with AdaBoost) and bagging (Random forests). The capacity management consumer applies validated models to evaluate alternative designs represented by solution candidates to the SPP. On the basis of predicted response times, SLA violations become calculable and resulting future scenarios can be analyzed with respect to their total costs, composed of operations costs and SLA penalties. Solution candidates which minimize the total costs are proposed in the capacity plan. It

summarizes the outcome of the PPSS method and provides executives with information required to make informed decisions.

3.4 Summary of the artifact's design

Passing through a requirements engineering process, functional and quality requirements compose the basis on which a new capacity management method for EAs was designed in order to address the identified research gap. The method is termed performance prediction supported service placement (PPSS) and involves two acteurs: One capacity management provider who maintains constructs and n capacity management consumers who apply the constructs. Main constructs of the process include placement algorithms, prediction models, and an application performance knowledge base. The former are used to solve a variation of the service placement problem which was formulated in accordance with the requirement specification. Here, four heuristics, two metaheuristics and two hybrid algorithms were instantiated. As part of the design phase, ten types of placement constraints were identified to possibly limit the individual solution space. While heuristics entail a considerable manual effort to reflect such constraints, metaheuristics can effectively incorporate them into the fitness function. The algorithm output describes optimized designs in the form of solution candidates. These are passed to prediction models for further evaluation. Here, the compliance with performance-related SLAs is estimated in order to enable computation of potential penalties. Having both operations and penalty costs available, a future scenario analysis allows to select most cost-effective solution candidates, depending on anticipated workload trends. As part of an ex ante evaluation step, performance models were validated during the design phase before they are put into use during ex post evaluation. Therefore, the role of the capacity management provider was taken in order to train models on the basis of shared performance counters from different implementations of a COTS EA. Data preparation, modeling, and model evaluation was exemplified using the most frequently used business transaction type of the SAP sales and distribution module. Here, mean response times per dialog step are predictable with a mean absolute percent error below 20%. Trained models, however, are limited to a particular transaction type as the prediction of transaction-independent service performance could not be accomplished by means of the tested machine learning techniques. Successfully evaluated models and algorithms are to be stored in a central knowledge base, hosted by the capacity management provider. It integrates all relevant data, algorithms, and models, required to carry out the descriptive and predictive analyses as part of the PPSS method.

4 Evaluation of the performance prediction supported service placement

This chapter describes ex post evaluation, carried out to investigate the designed artifact in use (Sonnenberg and vom Brocke, 2012). Therefore, the PPSS method is applied with a focus on the two main components: Service placement and performance prediction. First, field experiments are used to evaluate solution consistency, solution quality and efficiency of service placement algorithms across a variety of SPP cases, formed by real data (EVAL 3.1). For the prediction component, a qualitative approach is followed in order to study the artifact in-depth and evaluate the utility of the performance models. Hence, on the basis of real data, a case study is constructed around the artifact to allow for a scenario-based evaluation as proposed by Hevner et al. (2004). Here, in alignment with the overall research goal, results are further analyzed with respect to cost savings (EVAL 3.2). Finally, a prototypical implementation of the components, including the APM-KB, allows to apply parts of the artifact in a real business environment. Here, pilot users take the role of capacity management consumers and provide user feedback which helps to evaluate operationality and ease of use of the PPSS method along with its fidelity with real world phenomena (EVAL 4). First, however, ex ante evaluation steps, addressed throughout chapter 2 and 3, are summarized in the opening section of this chapter (EVAL 1 and EVAL 2).

4.1 Summary of EVAL 1 and EVAL 2

Evaluation activities on level 1 and 2 (cf. Table 1.1) are described throughout the chapters 2 and 3 in an order that follows the logical document structure.

With respect to the *relevance* and *importance* as part of EVAL 1, challenges and chances in the domain of capacity management for COTS EA were outlined from both scientific and practical perspectives. Optimization potential was derived from mean server utilization levels which typically account to not more than 10-20%. To address the resulting lack of energy efficiency, server consolidation was introduced to be an effective instrument. *Novelty* of the artifact, as discussed during the research gap analysis, lies in the extent to which specific EA characteristics are supported and utilized in order to design a cost-effective method for their capacity management. Particularly, the common usage of COTS software allows to leverage economies of scale by means of standardized performance models. Therefore, one *applicability* aspect of the artifact's design (part of

EVAL 2) was evaluated by means of a descriptive analysis across various implementations of a COTS EA with respect to the transaction usage. Dominating usage of standard functions foster the idea behind the designed method. Another applicability aspect was evaluated using benchmarking as part of performance model evaluation. As acceptable errors could be achieved, results contribute to the evaluation of the design's applicability. Finally, *(economical) feasibility* was ensured steadily through the intense involvement of industry representatives, i.e., during requirement elicitation, during the identification of suitable placement constraint types, or during the data preparation phase when constructing the performance models. The ex ante evaluation activities 1 & 2 embrace the design phase. After construction of the artifact, ex post evaluation activities on level 3 & 4 are to be conducted. These are described in the following.

4.2 Field experiments on multi-dimensional service placement (EVAL 3.1)

According to Hevner et al. (2004), mathematical problem formulations, as used in Section 3.2.1, allow to utilize optimization proofs as means to evaluate IT artifacts. Input to evaluation activities on level 3 are instantiated (constructed) artifacts. As this section is concerned with the evaluation of designed placement algorithms, these are to be instantiated to solve the SPP in a way that is consistent with the artifact's specification. In particular, the algorithms are expected to " [...] behave according to [their] purpose and scope" (Sonnenberg and vom Brocke, 2012, p. 395) as defined in the requirement specification (cf. Section 3.1.1). The quality of an algorithm is indicated by the quality of solutions the algorithm generates. Hence, solution quality, as defined in Section 3.2.2, is to be measured and compared across different algorithm instantiations. Furthermore, capacity savings and the compliance with placement constraints are important metrics to be measured according to the requirement specification. Efficiency, on the other hand, refers to the effort required to achieve a result. Hence, an additional metric which describes the run-time of each algorithm is given. In order to take also implementation effort into account, maintainability of algorithms is discussed either. Finally, to evaluate generality, algorithm performance is to be compared across a large number of cases. In order to reflect the aforementioned criteria, evaluation on level 3.1 is conducted experimentally using real data from a variety of environments. For this reason, the evaluation method is referred to as field experiments. Besides ex post evaluation, this section answers the research question RQ4, which aims to identify algorithms that are most suitable to solve multi-dimensional service placement problems within EA environments.

4.2.1 Descriptive analysis of field data

The field experiments are performed on 516 unique use cases. Each case represents a data center, operating a number of services on a number of servers, either on-premise or off-premise. A service is defined as an instance of a COTS EA, which may be an application

or a database instance that is to be placed on a physical server. One service cannot be distributed across multiple servers but different services may belong to a distributed EA comprising, e.g., a database instance and one or more application instances, each running on different servers. To summarize, six types of instances can be distinguished on the basis of the data:

- Productive application instance,
- Quality assurance (QA) application instance,
- Development application instance,
- Database of a productive application instance,
- Database of a QA application instance, and
- Database of a development application instance.

All data was collected using built-in software instrumentation facilities of the EA itself (cf. Section 2.1.4). Hence, any execution of a business transaction is being logged and associated with performance and workload metrics, which are stored in the file system. After a monitoring period of 7 to 65 days for each use case, the collected data was archived, cleaned and transferred to a relational database schema. This schema represents the data layer of an APM-KB which was constructed according to the design, specified in Section 3.1.3. This way, resource demands at a time (in terms of CPU and main memory) could be mapped to running services, enabling to derive workload profiles for each service on a weekly or daily basis from the historical data. In the following, statistical insights are provided as part of a descriptive analysis. This way, the cases used for experimental evaluation of the solution algorithms are introduced. Furthermore, the analysis helps to asses the volume and the variety of cases, needed to draw conclusions on the generality of the artifact.

As shown in Figure 4.1, the use cases vary from very small environments with a minimum of two services being operated on two servers to large-scale landscapes with the largest use case comprising 130 services on 59 servers. On average, 12.6 services are to be allocated to 7.9 servers and, in total, 6,885 services are to be allocated to 4,074 servers.

As indicated by the lower bound in Figure 4.1, the number of services to allocate is higher than the number of available servers in every use case. However, across all analyzed servers, a mean CPU utilization of only 18.72% can be computed based on theoretical peak demands. Those demands would be achieved if all hosted services concurrently claimed their peak workload found for a given time interval in the historical data on the server under study. Since the targeted design must prevent all servers from being overloaded at any point in time, these theoretical peak demands are used to build workload profiles for the running services and, therefore, can be used when comparing mean server utilization before and after the optimization procedure. In practice, however, such constellations rarely occur, hence, the actual mean server utilization across all measured points in time amounts to only 7.91%. This value roughly corresponds to the mean utilization level of 8.28%, which was gathered in Section 2.2.1 from an investigation of 13,332 servers.

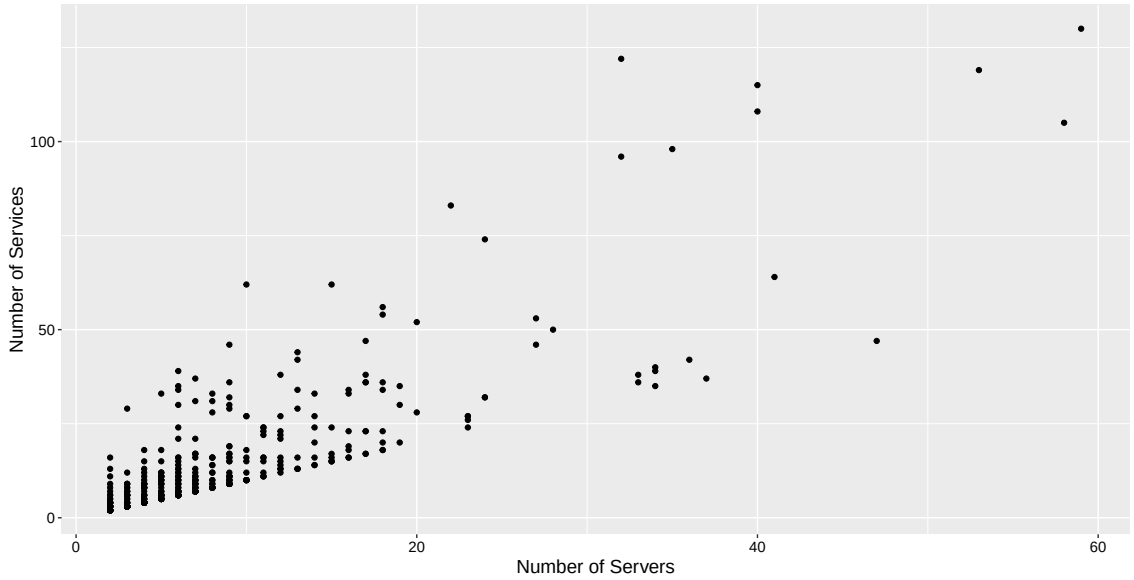

Figure 4.1: Number of servers and instances for the 516 investigated use cases.

However, it must be noted that the cases used for evaluation represent a subset of the data basis used in the referred investigation. In contrast to the Figure in Section 2.2.1, the histogram presented in Figure 4.2 shows the mean server utilization across all 516 cases on the basis of theoretical peak demands.

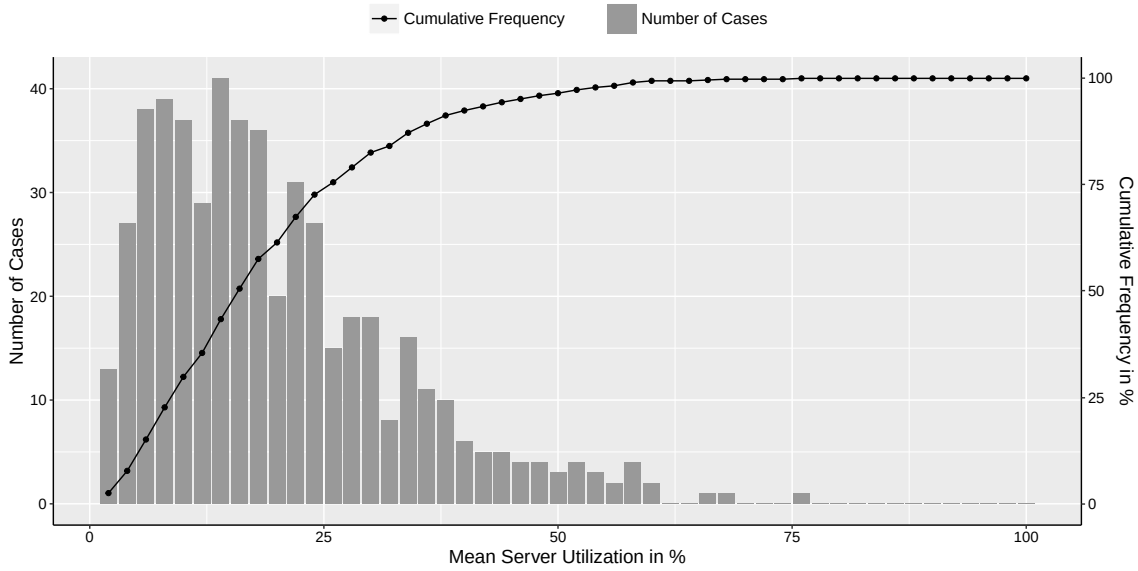

Figure 4.2: Histogram of mean server utilization using theoretical peak demands.

According to Figure 4.2, 75% of the cases show a mean server utilization of less than 28%. In half of the cases, the mean server utilization accounts to less than 19%. Actual (not theoretical) server utilization, as stated earlier, is even lower. In economic terms, the optimization goal is to avoid idling resources. Therefore, as stated in Section 2.2.1, mean actual server utilization levels around 65% should be targeted as a general rule of thumb,

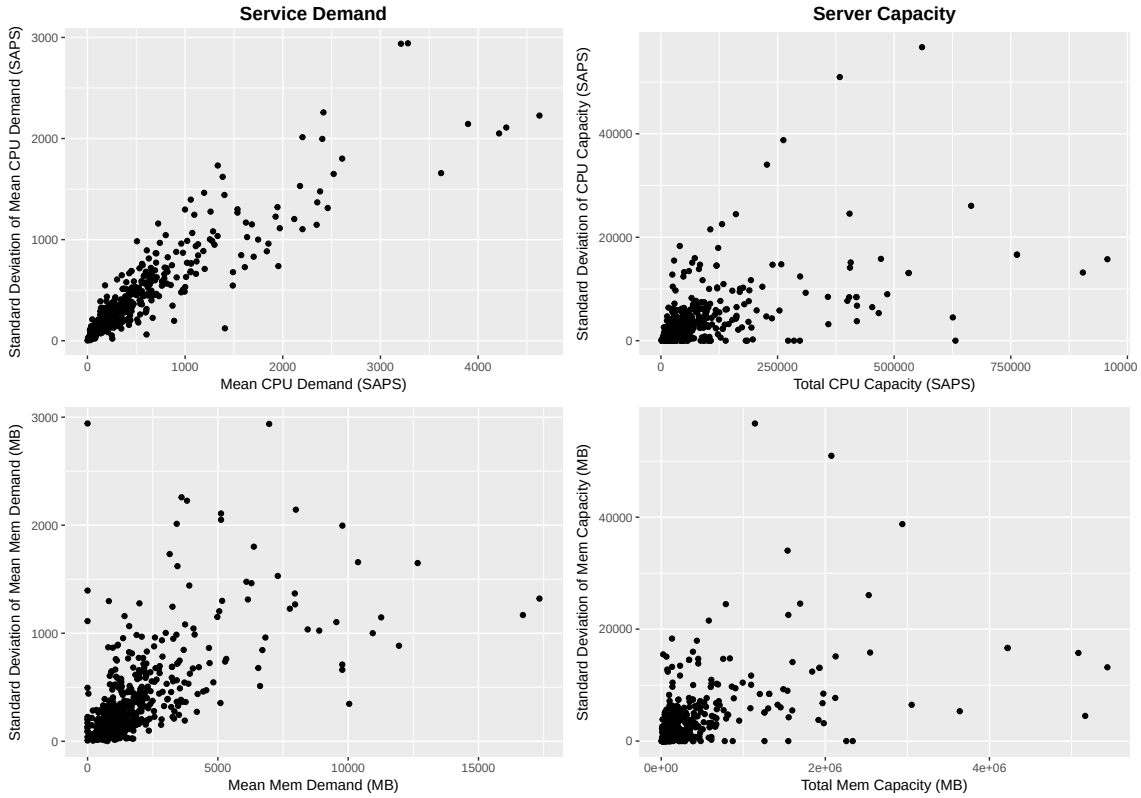

Figure 4.3: Mean values and standard deviations of service demands and server capacity.

subject to individual adjustment on the basis of business requirements. With respect to the variety of the cases, Figure 4.3 presents mean values and standard deviations of CPU and memory values for both service demands and server capacity. While main memory is measured in Megabytes, CPU capacity and demand can be measured in SAPS (cf. Section 2.1.1). In Figure 4.3, indicators for the complexity of the use cases are mean capacity demands and total capacity extents. On the y-axis, their respective standard deviation is provided. As illustrated on the left of Figure 4.3, cases with larger CPU workloads tend to be more heterogeneous than smaller ones while memory usage does not vary significantly with higher demands. On the other hand, the total capacity limit of a use case does not seem to be an indicator for server heterogeneity with respect to both resource dimensions.

The distribution of capacity limits and demands as well as the number of servers and services is illustrated in Figure 4.4. Herein, all box plots show a large scale and generally a high spread outside the interquartile range. The large number of outliers indicates a high variety within the tested application area of the solution algorithms. As an example, server types range from small-scale volume servers to high-end servers, according to the definition provided in Section 2.2.1. The mean CPU server capacity per use case shows an interquartile range of 3,364 to 10,356 SAPS with a median accounting to 5,987 SAPS. Across all use cases, the smallest server provides a CPU capacity of only 154 SAPS while the largest server can serve up to 178,524 SAPS. Therefore, the maximum throughput of

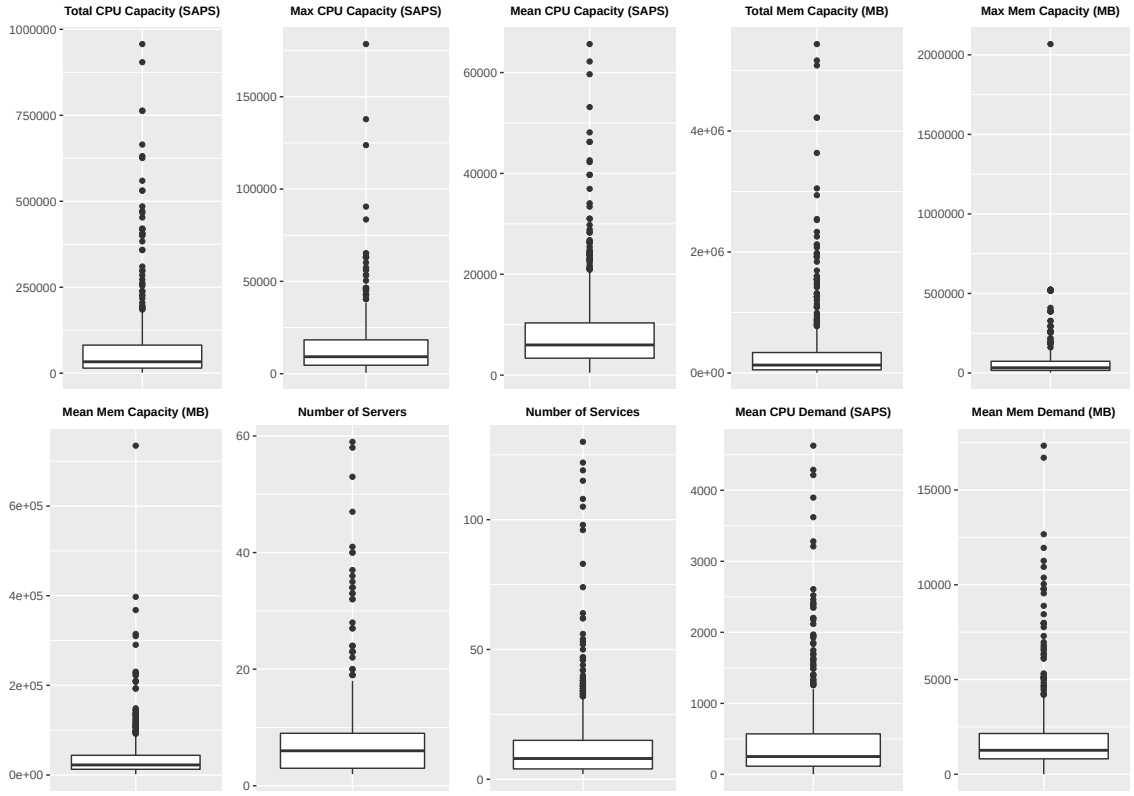

Figure 4.4: Descriptive metrics for the 516 investigated use cases.

the analyzed servers largely varies between a minimum of 3,080 and a maximum of up to 3.5 million order line items that can be processed per hour. Main memory for 50% of the servers ranges from 16.38 to 73.72 GB. The greatest outlier in the upper right chart of Figure 4.4 represents an in-memory database server providing 2 TB of main memory and 123,790 SAPS.

As further illustrated by Figure 4.4, idling services produce minimum demands of 0 SAPS respectively 0 MB. While the maximum CPU demand of a single service at a time accounts to 66,755 SAPS, a maximum memory amount of 124,798 MB was assigned to another service at a different time. The interquartile range of mean resource demands per case varies between 115 and 571 SAPS as well as 816 and 2,152 MB respectively. A summary of the described information, grouped by use case, can be found in Table 4.1.

Measure	**CPU capacity in SAPS**	**Mem. capacity in MB**	**CPU demand in SAPS**	**Mem. demand in MB**
Min	1,450	4,861	0	0
Max	957,086	222,728,412	66,755	124,798
Median	33,214	131,072	162	802
Mean	75,806	784,506	1,014	2,738

Table 4.1: Total case capacity and service demand across all investigated use cases.

4.2.2 Experiment setup

To cover a large variety of possible optimization scenarios, for each of the introduced use cases (cf. Section 4.2.1), three scenarios with different consolidation pressure are defined. It was argued in Section 3.3.4 that consolidation pressure can be adjusted over the maximum server utilization constraint type C_{util}. Therefore, the following three scenarios were designed:

- **Scenario UC:** Unconstrained scenario,
- **Scenario C85:** Constrained scenario with $C_{util} = 85$,
- **Scenario C65:** Constrained scenario with $C_{util} = 65$.

In the unconstrained scenario, the maximum server utilization translates to 100%. This scenario therefore represents a high risk attitude of the decision maker with respect to the probability of server overloads. Scenario C65 represents the generic yet domain-specific recommendation, provided by Janssen and Marquard (2007) and therefore a rather low risk attitude which is based on prevalent sizing guidelines. Finally, a medium risk attitude is covered in Scenario C85 whose maximum server utilization lies between the recommendation and scenario UC, which aims at maximum savings. As the scenarios C85 and C65 invoke constrained instantiations of the SPP, a set of realistic placement constraints was defined in addition to the maximum server utilization. It was designed with the objective to cover real-world restrictions of consolidation efforts on the one hand, and to allow for comparability of algorithms on the other hand. The following list summarizes the set of constraints for C65 and C85:

- For all databases of productive instances, an isolation constraint C_{iso} is defined.
- For the largest 20% or the largest two (higher value) productive application instances, in terms of their peak CPU demand over a day, an anti-colocation constraint C_{antico} is defined.
- For each productive application instance, an extra resource constraint $C_{extrares}$ is defined which demands 10% extra CPU capacity.
- For all servers, a maximum server utilization constraint C_{util} is defined with either 85% or 65% of the servers' total capacity that should not be exceeded.

To compare algorithmic performance, each of the eight solution algorithms (FFDmax, FFDsum, BFDmax, BFDsum, GA, GGA, GA_FF, GA_BF) is used to solve each scenario, applied on each of the 516 use cases. In other words, every use case is solved 24 times (3 scenarios and 8 algorithms). In total, 12,384 experiments are conducted. All solution algorithms are implemented and executed using the JAVA development kit (JDK) in version 8. The heuristics do not require any parameters to be obtained. Suitable parameters for metaheuristics were derived from sensitivity analyses using a subset of four exemplary cases. These parameters, which are explained in Section 2.2.4, are presented in Table

Algorithm	$maxGen$	Selection	μ	λ	Elitism	p_{mut}	t	p_t
GA	200	comma	100	200	1	0.1	4	0.9
GGA	75	comma	50	100	1	0.1	N/A	N/A
GA_FF/BF	50	plus	25	25	no	1	4	0.9

Table 4.2: Algorithm parameter set used for evaluation experiments.

4.2. As the GA and the GA_FF/BF apply tournament selection, tournament size t and selection probability p_t are provided in these cases. In contrast, the GGA uses stochastic universal sampling, where the number of pointers equals μ (cf. Section 2.2.4). Due to the randomness inherent to genetic algorithms, all metaheuristic and hybrid algorithms were executed in ten iterations and mean values are provided as experiment results. As stated earlier, it is the goal of the experiments to evaluate algorithms with respect to utility, efficiency, and generality. Therefore, the following metrics are captured for each experiment:

- Total capacity (CPU and memory) required by the computed design,
- Solution fitness (quality) of the computed design,
- Resource overflows (CPU and memory),
- Penalty factors for each placement constraint violation,
- Mean server utilization within the computed design, and
- Algorithm run-time.

On the basis of these metrics, additional evaluation metrics such as mean capacity savings or the distance to best performing algorithms can be computed and grouped by algorithm type. Furthermore, optimized designs may be compared to existing designs in order to evaluate economical feasibility as also done by Gmach et al. (2008). This way, algorithms and their suitability with respect to different variations of the SPP are evaluated in the subsequent section.

4.2.3 Experiment results

As IT artifacts are to be evaluated using relevant quality attributes (Hevner et al., 2004), solution fitness is used as the main quality metric for evaluation. As described in Section 3.2.4, solution fitness initially corresponds to $-TOTALSAPS$ and should therefore be maximized by the algorithms. The value is then subtracted (penalized) by all resource overflows, measured in SAPS, and by constraint penalty factors. Hence, aspects which reflect the applicability of solutions are already inherent to the solution fitness. In order to make this case-dependent metric comparable across all experiments, first, the best fitness value fit_{best}, achieved by any of the algorithms, was identified for each use case and scenario. Second, the relative distance fit_{diff} to this value was computed for each algorithm, as expressed in equation 4.1.

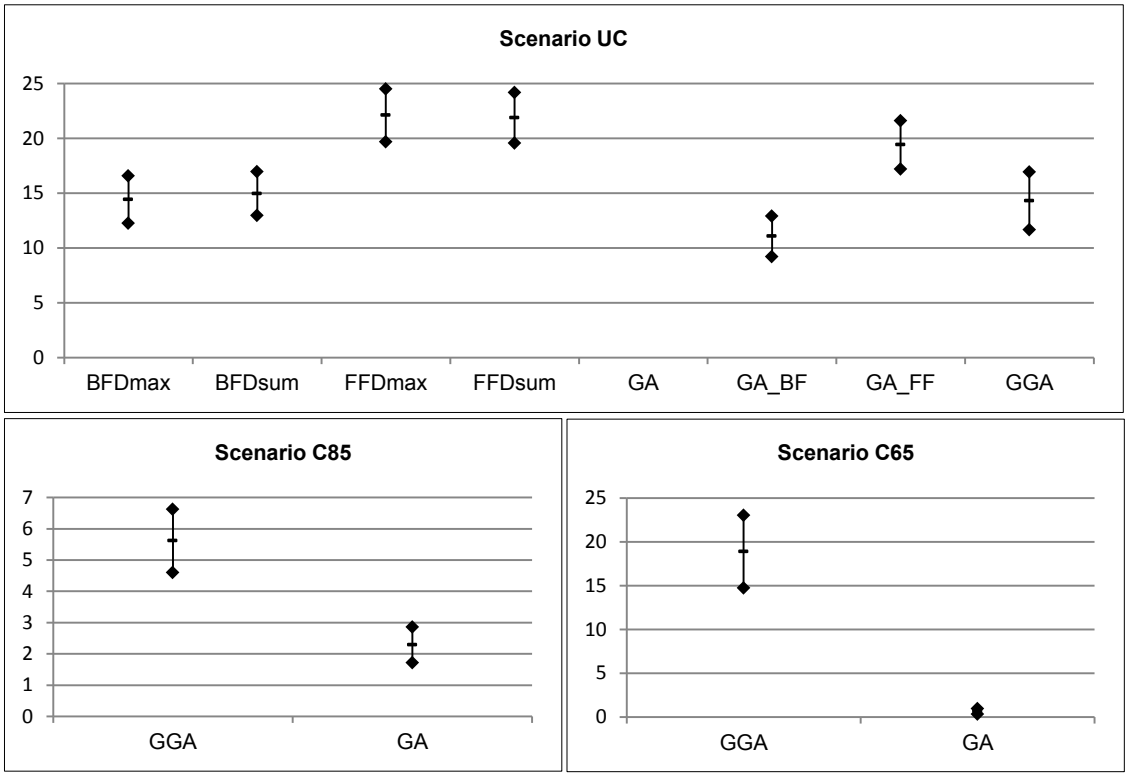

Figure 4.5: Distance to best fitness value for each algorithm and scenario.

$$fit_{diff}(alg) = \frac{fit(alg) - fit_{best}}{fit_{best}} \tag{4.1}$$

If fit_{diff} of a particular algorithm accounts to 0, this algorithm computed a solution with the best quality compared to all other algorithms for this case and scenario. In general, the lower the value of fit_{diff}, the better the algorithm with respect to the defined solution quality. When grouping the results by algorithm, mean values of fit_{diff} indicate algorithm suitability. Furthermore, the 95% confidence interval was computed, referring to the lower and upper bounds which encompass the true value in 95% of the cases. Across all cases, results for the mean fit_{diff} and its confidence intervals are shown in Figure 4.5 for each scenario and algorithm. In general, heuristics perform well in unconstrained scenarios while they fail to reflect placement constraints. In fact, mean values of fit_{diff} for pure heuristics differ between 174.73% and 552.46% across the constrained scenarios and, therefore, were not included at the bottom of Figure 4.5. In the unconstrained scenario, best-fit heuristics generally deliver solutions of higher fitness than first-fit ones while the reference value for sorting (max or sum) does not seem to make a significant difference. Best suitable algorithms for unconstrained scenarios are the GA_BF (11.08%) and the GGA (14.32%). A pure GA, however, shows a mean difference to the best fitness value of 317.48% with a confidence interval ranging from 277.67% to 357.23%. This behavior is assumed to result from the different encoding strategies of the GA and the GGA. In general, any kind of genetic algorithm performs well if small changes in a generation

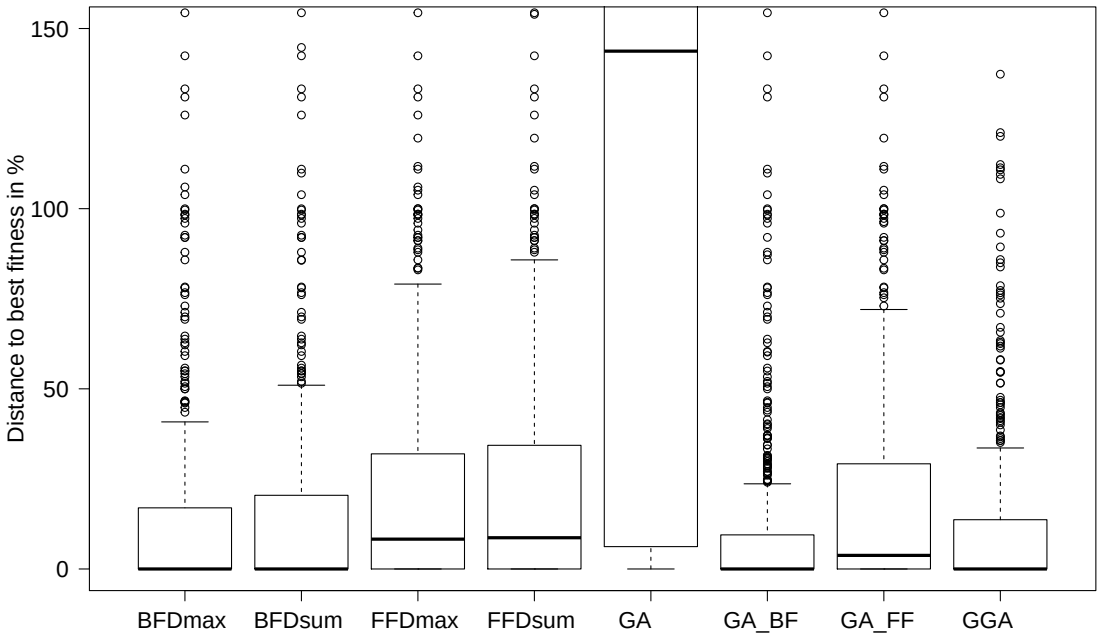

Figure 4.6: Distance to best fitness value for each algorithm in the unconstrained scenario.

result in small changes of the fitness so that the search process is guided on the basis of the fitness function. The GGA uses a grouping representation which fits the problem of grouped services on servers very well while the GA, due to its unnatural representation, too often seems to fail in improving the solution fitness iteratively. In the scenarios C85 and C65, heuristics perform poorly due to their lack of intentional constraint consideration. Here, hybrid approaches (GA_FF and GA_BF) perform much better with mean values of 79.03-84.44% (C85) and 266.07-242.06% (C65) where the BF always performs better than the FF. Hybrid approaches, however, are limited to the power of heuristics which employ the actual allocation. The GA-parts, in these cases, are designed to find a new ordering of the input sequence but cannot benefit from improvements over generations since they depend on what the FF or BF allocate which implies the risk of irrational changes of the fitness. In this sense, the algorithms try to find randomly an ordering that fulfills the constraints. Solutions to constrained scenarios found by the GA and the GGA are better by magnitudes: The GA achieves mean value of 2.29% in C85 and 0.63% in C65. The GGA achieves mean value of 5.61% in C85 and 18.89% in C65. Hence, the scenario C65 reveals an increasing superiority of the GA with harder constraints when compared to the GGA. Finally, the confidence intervals are generally greater with worse fit_{diff}. Confidence intervals, however, do not show the actual data distribution, e.g., how frequently which algorithm performed well. The box plots in Figure 4.6 allow for more specific interpretation of the experiment results. As illustrated in Figure 4.6, every algorithm performed best in at least one case since the extreme of the lower whisker accounts to 0 for every algorithm. Also the lower hinge (representing the first quartile of the data), except for the GA, accounts to 0 for each algorithm, saying that every algorithm was able to achieve best values in at

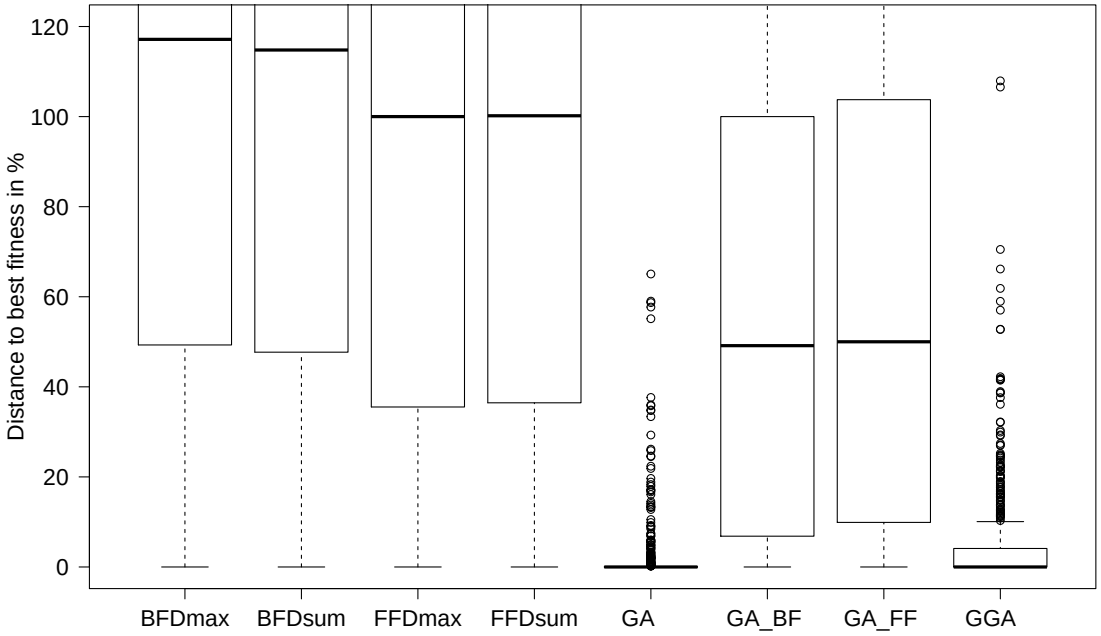

Figure 4.7: Distance to best fitness value for each algorithm in constrained scenario C85.

least 25% of the cases. Furthermore, the best-fit heuristics, the hybrid algorithms and the GGA performed best in at least 50% of the cases as represented by their median. By far, the pure GA performs worst in the unconstrained scenario with a median of the distance to the best fitness value accounting to 143.72% and the extreme of the upper whisker accounting to 1032% which reaches out of the scale in Figure 4.6. In the constrained scenario C85 (cf. Figure 4.7), similarly, every algorithm performs best in at least one case. This comes as a surprise as heuristics consider constraints only randomly. More reliable constraint compliance, however, is provided by the GA and the GGA which achieve best fitness values in at least 50% of the cases. Here, the GA outperforms the GGA significantly and achieves best values even in three quarters of the cases as represented by the upper hinge accounting to 0. According to Figure 4.6, experiments in which the GA does not achieve the best result can be considered as outliers. The greatest outlier is represented by a maximum fitness difference of 65.05% to the best algorithm. The GGA, in contrast, achieves a maximum fitness distance of 4.11% in 75% of the cases which represents a considerable result too. Figure 4.8 reveals a similar tendency for the constrained scenario C65. Here, as indicated by the different scales of the two charts, differences between the algorithms roughly doubled. Again, all algorithms achieved best values in at least one case. Solutions of highest quality are produced by the GA and the GGA. The GA performs best in at least 75% of the cases and fails only in exceptional cases as even its upper whisker accounts to 0. The GGA achieves best values in at least 25% of the cases. Its median accounts to 1.21%, telling that for half of the cases, the difference to the best value is lower than 1.21%. In three quarters of the cases, the percentage difference to best fitness values is below 14.93%. Further data on the solution fitness are given in Table B.2 (Appendix

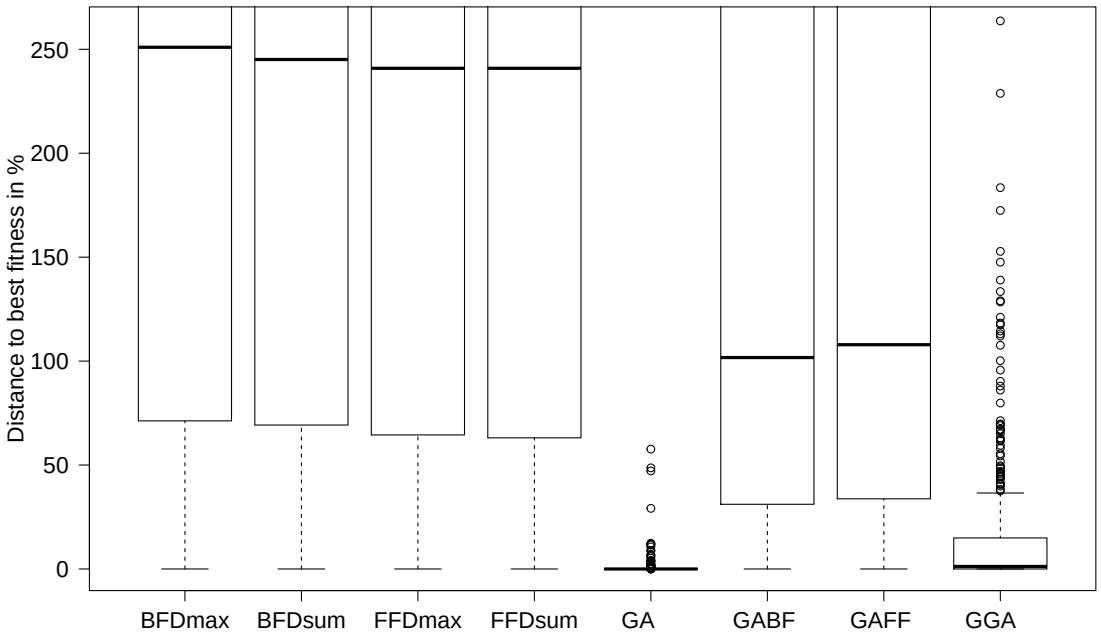

Figure 4.8: Distance to best fitness value for each algorithm in constrained scenario C65.

B). Here, mean values and the confidence intervals for fit_{diff} are provided. In order to investigate the differences between the GA and the GGA in more detail, the components of the solution fitness must be analyzed individually. As stated in Section 3.2.2, the fitness value compasses the required server capacity and all constraint penalty factors. Therefore, after investigation of $diff_{fit}$ it remains unclear, whether the GGA suffers from unsaved capacity or from constraint violations when compared to the GA. To address this issue, Figure 4.9 shows, for scenario C85, the two fitness components separately. Similarly to the fitness evaluation, distances to best achieved values are shown. As can be seen on the left hand side, the distribution of savings is very similar when comparing the GA (median of 0.75%) and the GGA (median of 0.74%). In terms of constraint compliance,

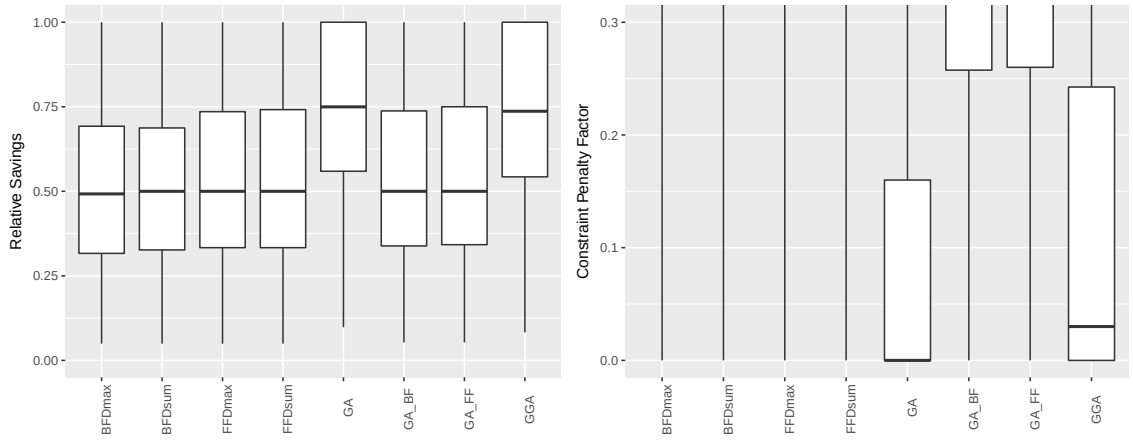

Figure 4.9: Relative savings and constraint penalty factors in constrained scenario C85.

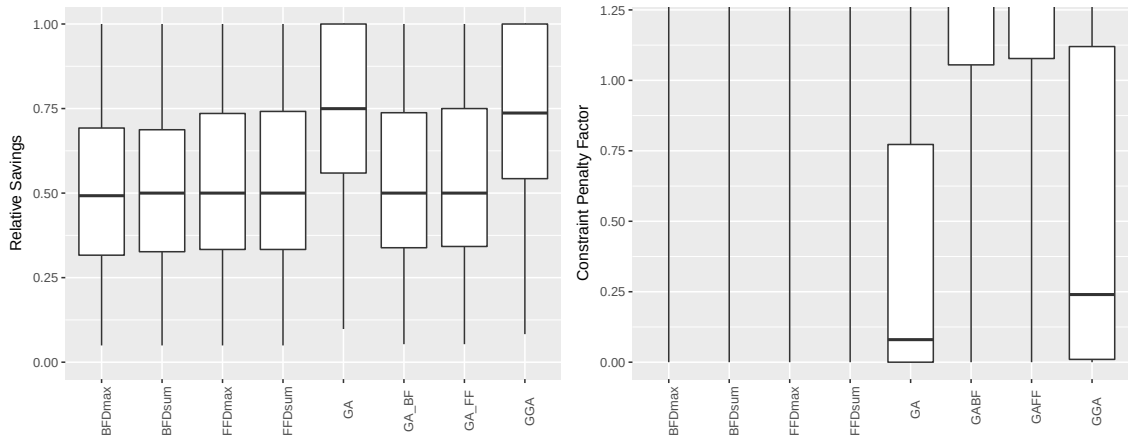

Figure 4.10: Relative savings and constraint penalty factors in constrained scenario C65.

however, the GA outperforms the GGA. Here, only the GA shows a median of 0. The upper hinge indicates that the GA achieves a maximum difference of 0.16% to the best penalty factor in 75% of the cases. This distance results to 0.25% for the GGA. In scenario C65, this relation becomes even more clear (cf. Figure 4.10). Hence, the GGA fails to consider certain types of constraints when compared to solutions found by the GA. In particular, the isolation constraint which was defined for productive database services, and the maximum utilization constraint are violated significantly more frequently by the GGA when compared to the GA. An explanation to this can be found in the alternative ways of encoding a solution (cf. Section 3.2.4). While the mutation operator produces equal results for the GA and the GGA, recombination is performed differently. If, in case of the GA, two parents which fulfill the isolation constraint for an instance, are recombined, the child will also fulfill this constraint. In contrast, the recombination procedure of the GGA (as explained in Section 3.2.4) may lead initially to interim states of lost instances or instances which are allocated twice. The subsequent repair mechanisms delete duplicate instances and insert lost instances. This way, instances which were already isolated in both parents, may get a new neighbor due to the injection. Furthermore, the injections may introduce overflows and, thus, violate targeted utilization levels as defined by the maximum utilization constraint. Further details on the constraint compliance are given in Table B.1 (Appendix B) which presents mean penalty factors for each algorithm.

To conclude the fitness evaluation, a rather straightforward quality measure is shown by Figure 4.11. Here, it is counted how often an algorithm achieved the best fitness value within the same scenario and use case. If multiple algorithms achieved this value, all of them are counted. Therefore, within each scenario, the total frequency sums up to more than the 516 investigated cases. Figure 4.11 confirms the “No-free-Lunch-Theorem” of Wolpert et al. (1997) according to which an optimization algorithm that performs well for one problem class must fail in another one. For example, while the GA solves constrained scenarios extraordinary well, this algorithm is to be declined for unconstrained scenarios. The GGA, however, shows relatively reliable behavior across all scenarios and is

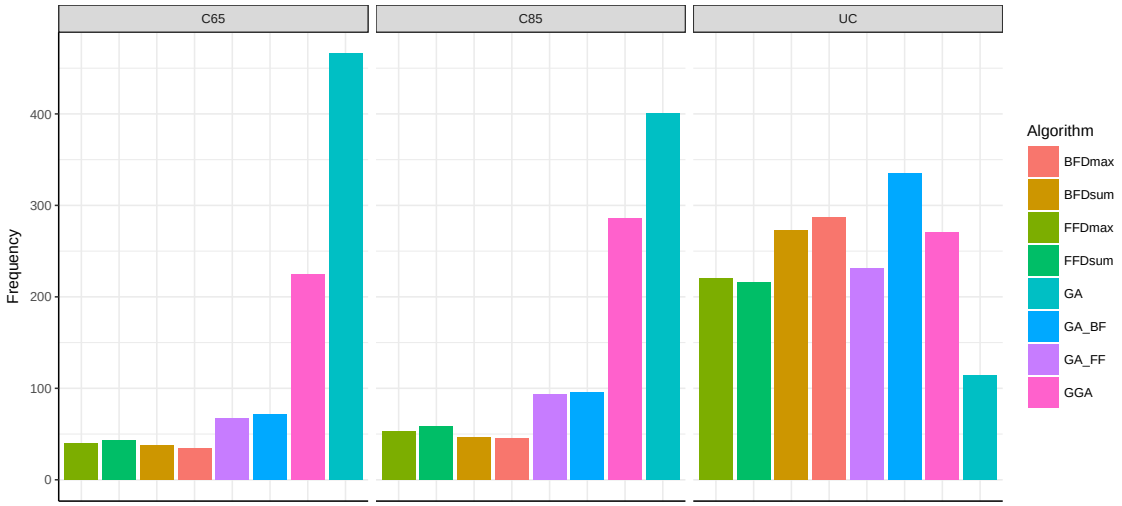

Figure 4.11: Frequency of best-solved experiments for each algorithm and scenario.

the recommended algorithm of choice if an all-purpose solution is needed. More granularly, unconstrained problem classes may be solved preferably by the hybrid algorithm GA_BF and constrained scenarios are well suitable for a pure GA. Heuristics are only applicable in unconstrained scenarios. Here, best-fit heuristics generally outperform first-fit heuristics and, interestingly, the GA and the GGA.

In the targeted domain, run-time is not a prevailing issue, since solution candidates are computed and evaluated offline. Nevertheless, algorithm run-time was measured since results may also be of interest for approaches which optimize resource allocations online. Mean results for each scenario are provided in Table 4.3. Associated confidence intervals are given in table B.2 in Appendix B. Clearly, results are subject to hardware and software characteristics of the machine running the experiments. However, values provide a rough magnitude and allow to compare algorithm performance. In the chosen experiment setting, no algorithm ran longer than 2.3 seconds. Therefore, all algorithms meet quality requirement Q3, according to which, due to the form of service delivery, solutions must be found within seconds. Heuristics compute solutions highly efficiently and are well suitable for unconstrained online approaches. With respect to metaheuristics, the GGA consumes

Algorithm	**Scenario UC**	**Scenario U85**	**Scenario U65**
FFDmax	0.258	0.225	0.218
FFDsum	0.130	0.215	0.203
BFDmax	0.076	0.285	0.269
BFDsum	0.072	0.215	0.233
GA	860.207	2,219.821	2,206.973
GGA	244.264	500.506	494.452
GA_FF	265.151	369.195	363.513
GA_BF	276.130	376.931	381.080

Table 4.3: Mean run-time per algorithm in ms.

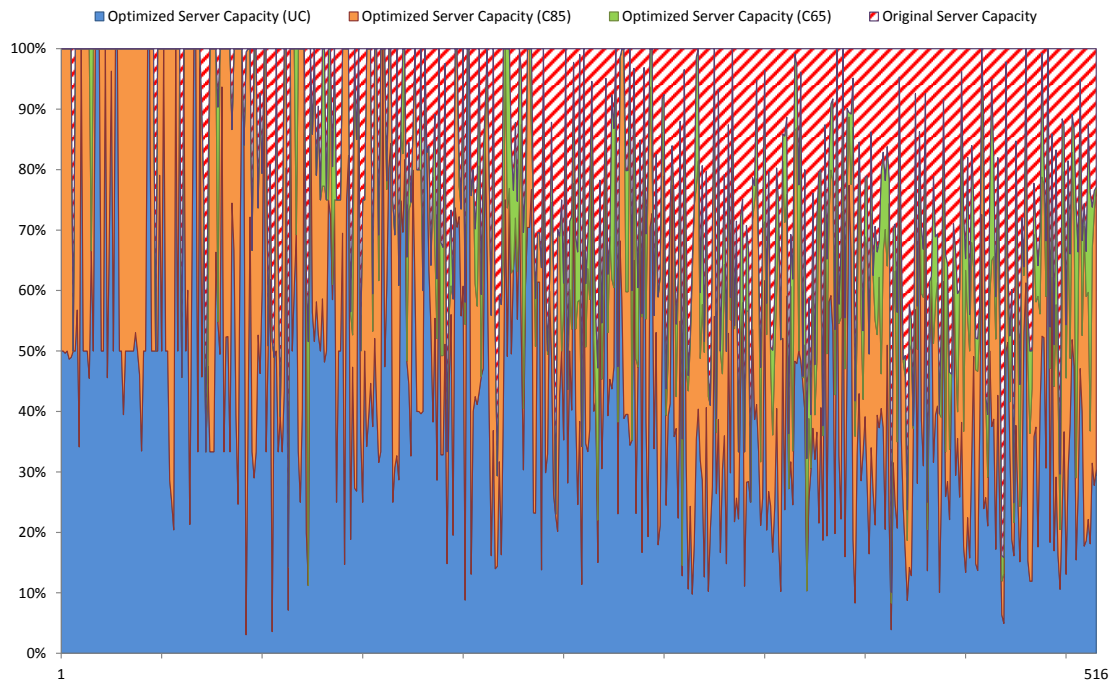

Figure 4.12: Capacity savings across best solutions for each use case and scenario.

not more than 25% of the time needed by the GA and, therefore, may also be suitable to complement online optimization approaches. In general, constraints limit the search space as the number of possible solutions is reduced. On the other hand, constraints increase the complexity of the solving process as the fitness function is extended by constraint evaluation. For this reason, a decelerating effect on the run-time can be observed when constraints were modeled.

Contribution to research question 4
Unconstrained scenarios can be solved efficiently by heuristics, hybrid approaches, and a grouping genetic algorithm in less than one second. Constrained scenarios, on the other hand, benefit from a guided search process on the basis of a fitness function which penalizes resource overflows and constraint violations. Here, a genetic algorithm achieved best solution quality, followed by a grouping genetic algorithm. The latter serves as a suitable all-purpose technique, if a single algorithm is to be selected for unknown problem instantiations.

Finally, independent of the algorithm type, economical feasibility is to be evaluated in accordance to quality requirement Q1. Therefore, mean savings and mean utilization levels are analyzed in the following. In order to evaluate saving potential, solutions of best fitness were analyzed for each use case. In this regard, Figure 4.12 shows the capacity requirements for each use case and scenario. Here, 100% refers to the original capacity and, therefore, represents the reference value to any saving potential. After optimization, the red hatched area can be saved in any scenario, including C65. In scenario C85, the red hatched and the green area are to be eliminated. The blue area represents the

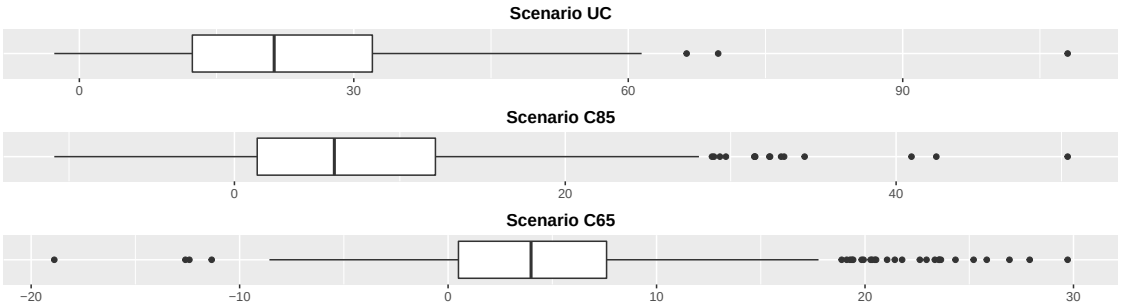

Figure 4.13: Mean distance of CPU utilization from original to optimized design.

percentage of original capacity that is needed if no constraint applies, resulting in the highest saving degree. The use cases in Figure 4.12 are sorted by complexity (given by the number of possible solutions) in ascending order. Complexity varies from 4 to 1,62465635425464E+230. This way, it becomes clear that more complex cases provide higher relative saving potential than small cases. This coherence can be explained by economies of scale or simply by the fact that small environments are still manageable in a traditional manual manner while capacity management becomes more complex with larger environments. Hence, use cases of low complexity may already represent optimized designs. For the analysis, only solutions with best possible constraint fulfillment were considered as fit_{diff} accounts to 0. In the rare cases of solutions in scenario C65 requiring less capacity than in C85, C85 values were adapted to the value of C65 since a solution which fulfills C65 also fulfills C85 in any case. To summarize, the following savings are achievable for the investigated use cases:

- Scenario UC, in average, requires 44% of the original server capacity.
- Scenario C85, in average, requires 72% of the original server capacity.
- Scenario C65, in average, requires 78% of the original server capacity.

Hence, mean savings of at least 22% are achievable for any scenario, including C65 which complies with the recommended level of server utilization in the target domain. If the avoidance of overflows is the only objective and no additional constraints are modeled, as done in the majority of the studied related work, 56% of the original server capacity can be eliminated. Further details on mean savings per algorithm and scenario are provided in Table B.2 in Appendix B. To conclude, the approach meets quality requirement Q1 according to which the artifact is relevant economically if at least 30% of the original capacity can be saved in unconstrained scenarios and not less than 20% if constraints apply. Self-evidently, mean server utilization increases if a portion of capacity is removed. In order to compare utilization levels before and after the optimization process, as stated earlier, theoretical peak demands are used. On this basis, the mean CPU utilization across all observed cases amounted to 18.72% before optimization. Figure 4.13 shows the mean CPU utilization distance from the original design to the solution of highest fitness across all cases and all algorithms. For the unconstrained scenario, best performing algorithms

identified solutions of averagely 22.66% higher CPU utilization when compared to the original design. As expected, realistic constraint sets reduce the addressable optimization potential significantly, mainly determined by the maximum utilization constraint. Consequently, in scenario C85 respectively C65, mean utilization growth of 7.85% and 5.14% was obtained. However, even in constrained scenarios, considerable optimization results could be achieved with a maximum utilization growths of 50.37% and 29.72% respectively. In general, due to the non-proportional energy consumption of today's server architectures, even small changes in the levels of utilization increase overall efficiently and contribute to the economical relevance of the results.

4.2.4 Summary of EVAL 3.1

The implemented heuristics, genetic algorithms, and hybrid approaches were evaluated on the bases of three scenarios, using real monitoring data from 516 environments. In unconstrained scenarios, heuristics provide feasible solutions within short time (less than 0.3 ms). Here, the GA is outperformed by any other algorithm. If, however, a GA is used to optimize the input sequence to a best-fit heuristic (GA_BF), this algorithm performs best in terms of solution fitness if no constraints apply. Fitness was calculated on the basis of the required capacity, the avoidance of overflows, and constraint compliance. In contrast to the unconstrained scenarios, the GA achieves best results with increasing restrictions. Accordingly, the GA is the preferred algorithm if the solution space is limited by a number of constraints. In these cases, heuristics fail to compute solutions of sufficient fitness as their ability to consider constraints is limited to chance hits. Costly repair mechanisms, in terms of data pre-processing and solution post-processing, would be required in order to achieve acceptable levels of constraint compliance. Moreover, individual adaptions are to be implemented for each type of constraint which, according to Hermenier et al. (2013), are subject to change. Hence, due to their limited maintainability, heuristics do not comply with quality requirement Q4. The optimization potential generally increases with growing problem complexity. This fact suits the artifact's design well as large environments, at the same time, show lower customization degrees (cf. Section 2.1.1). If an all-purpose technique is desired, the GGA is to be recommended as it provides feasible solutions across all scenarios.

4.3 Case study on solution candidate performance prediction (EVAL 3.2)

This section addresses the secondary objective of PPSS. Here, solution candidates to the SPP are to be evaluated with respect to their total costs. Total costs comprise operations costs c_o and penalty costs c_{sla}. While the former is to be obtained on the basis of known capacity requirements, the latter implies performance estimations. Required performance models were evaluated quantitatively in Section 3.3. In the following, their utility in a particular business context is evaluated by means of a scenario-based demonstration.

4.3.1 Case description and prediction input

The case study is based on real data which comes from the same data set that was used to generate the field experiment data in Section 4.2.1. The application context is based on input provided by one of the pilot users of the method [1]. The input, however, was modified to protect sensitive business data and to support the demonstration purpose. Therefore, realities, as suggested by Sonnenberg and vom Brocke (2012) for EVAL 3 activities, are given. The studied case describes a large retail company whose IT resources are grouped in multiple German data centers, each hosting COTS EAs. One of the data centers is facing a centralization project which entails the integration of additional markets into the existing EA system landscape. According to the project plan, the number of served markets increases successively. For example, on the way from the current state to the

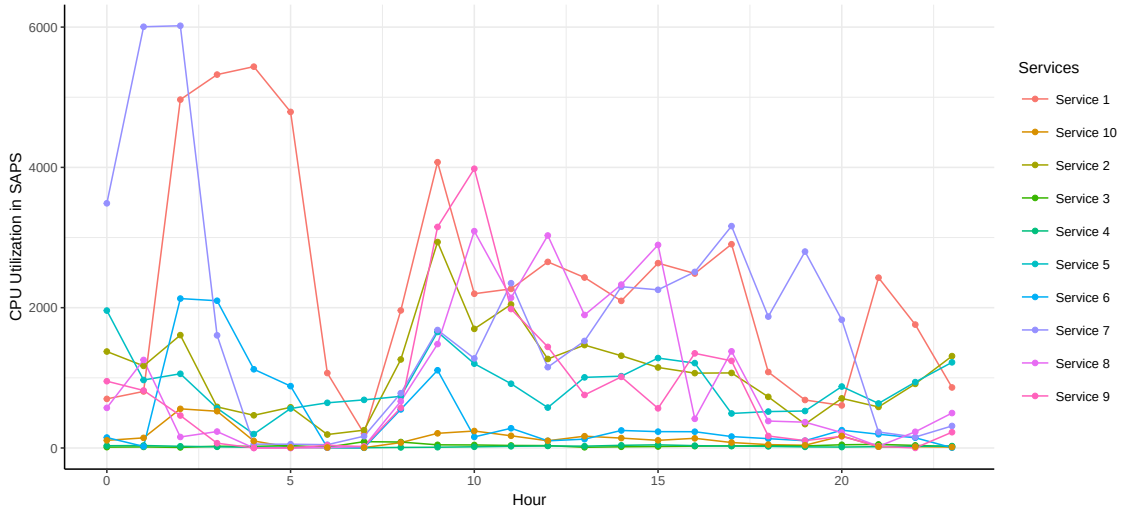

Figure 4.14: CPU workload of the ten services.

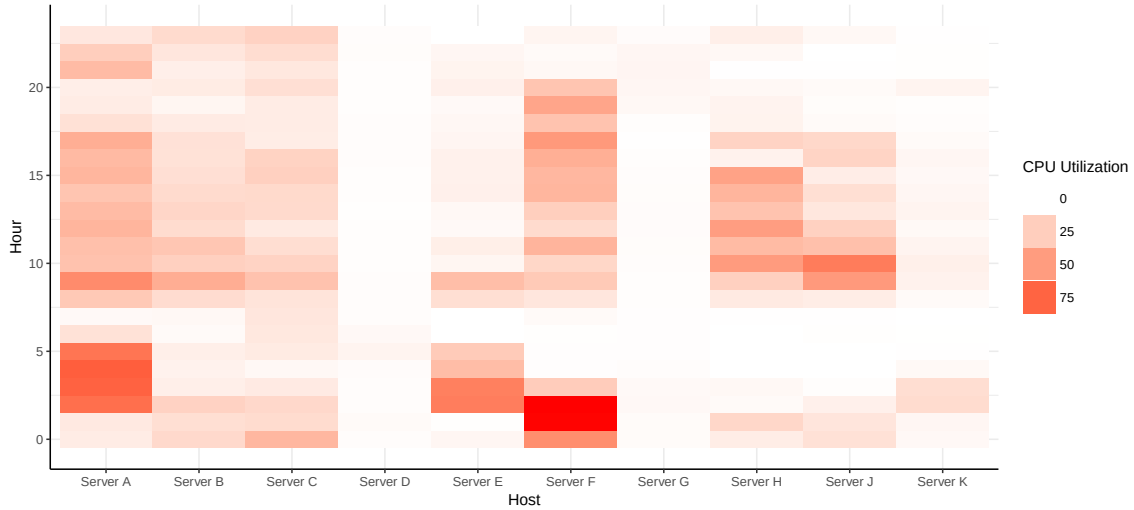

Figure 4.15: CPU utilization of the ten servers.

[1] Telephone interview at March 13, 2017

first milestone, the number of markets will have doubled. In the future, however, the business strategy prescribes to add about 125% of the current number of markets. Hence, workloads are planned to increase in the future and the decision maker is interested to estimate the respective effects on the design. In its current state, the data center hosts ten services. Due to the prevailing provisioning practice, which involves to procure new hardware for new services, ten servers are available in total although two of them show extra-ordinary low utilization. The workload profiles for the running services and the utilization levels for the servers are depicted in Figure 4.14 and 4.15. The actual mean CPU utilization across the whole environment accounts to 14.30% and all servers provide a total CPU capacity of 68,369 SAPS.

The data shown in Figures 4.14 and 4.15 is used as input to the SPP. As outlined in Section 3.3.4, three different target values were defined as maximum utilization levels in order to control the consolidation pressure. For this purpose, the maximum utilization constraint was utilized. As a result, three design alternatives represent three solution candidates of different risk attitudes. In accordance with the evaluation results presented in Section 4.2.3, the GGA was used to produce solution candidates. These are presented in Table 4.4.

In order to evaluate the solution candidates with respect to their performance in varying load situations, load factors were defined in accordance with the business strategy. Hence, load factors from 1 to 2.25 scale the existing values of the workload metrics for each time interval. As a result, six variations of the workload profiles, depicted in Figure 4.14, are used to feed the performance model, resulting in a total number of 18 future scenarios (six for each design). If probabilities for the alternative load factors are available, design recommendations, with the objective to minimize the total costs, can be provided. Otherwise, results are to be discussed together with decision makers. For the 18 future scenarios, mean response times per dialog step for the business transaction to change sales orders (transaction code: VA02) of medium complexity are estimated. Here, one of the model types evaluated in Section 3.3.2 such as Random forests, SVM with RBF kernel, or AdaBoost can be applied. To demonstrate the given use case, Random forests were used due to their existing integration with the analysis layer of the APM-KB at the time of model selection. In the case under study, 587 measured hours show at least one execution of the investigated business transaction. Response times were estimated as part of the step *Predict service performance* of the PPSS method. In the following,

	Design A	**Design B**	**Design C**
Maximum utilization level	100%	65%	45%
Number of servers	5	7	8
Total CPU capacity in SAPS	30,846	51,182	59,126
Hardware savings	high	medium	low
Risk attitude	high	medium	low

Table 4.4: Computed solution candidates.

prediction results are analyzed for the introduced use case in order to demonstrate the utility of the method. Therefore, the following activities are carried out as part of the step *Analyze future scenarios* and, finally, lead to decision support in terms of recommended designs. These findings contribute to the *Capacity plan* which represents the outcome of the process.

4.3.2 Prediction results and operations costs

All prediction results were archived together with the input data in order to allow for subsequent analysis. The initial yet unsurprising assumption of higher loads causing higher response times is generally confirmed by the results. As shown by the box plots in Figure 4.16, response times can be improved by lowering utilization rates. This measure, however, typically requires greater capacity. As indicated by the value distribution in Design C,

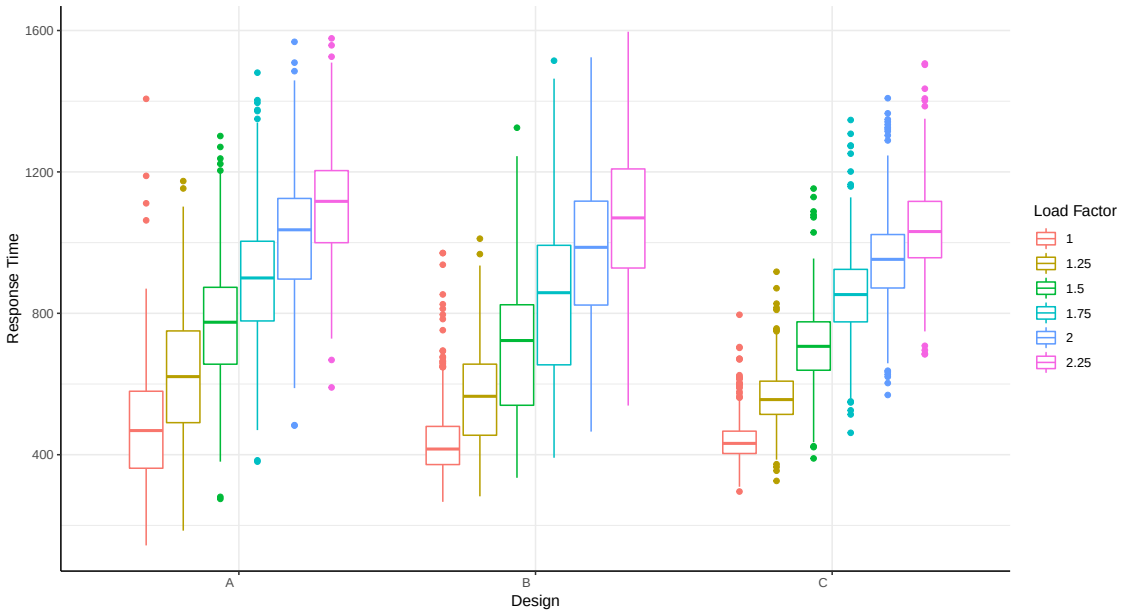

Figure 4.16: Distribution of mean response times per dialog step for each future scenario.

capacity buffers seem to effectively catch load peaks, resulting in a smaller interquartile range and a closer bound for outliers. Hence, this design reduces the variance and tend to be the most stable option in terms of performance. If deadline constraints require the response times to be always below a certain threshold, the illustration in Figure 4.16 allows to draw a respective line and limit the solution space to scenarios which do not exceed this line.

Having both performance estimations and capacity requirements in place, total costs can be calculated. One component of the total costs is represented by the operations costs c_o. This item is subject to a number of aspects which vary for each organization. However, for the sake of demonstration, the computed designs were translated into costs in US Dollar using the "AWS Simple Monthly Calculator"(Amazon, 2019). This tool distinguishes server classes which can be associated to domain-specific metrics such as

Server class	SAPS range	Operations costs in $	Used in Design A	Used in Design B	Used in Design C
1	1 – 2,379	146.4	-	-	-
2	2,380 – 4,758	218.87	-	1	1
3	4,759 – 9,515	364.54	5	4	5
4	9,516 – 19,030	655.88	-	2	2
5	19,031 – 37,950	1237.82	-	-	-
		Operations costs:	1,822.7 $	2,988.79 $	3,353.33 $

Table 4.5: Operations costs for the computed solution candidates.

SAPS. For this purpose, a mapping table, provided by Amazon in Alvarez (2015), was used. Table 4.5 shows the resulting costs for each server class and indicates how often each server type was used by a design. The resulting total operations costs are calculated for each design. This exemplary cost calculation leads to the operations costs c_o in USD. In the following, penalty costs c_{sla} are to be estimated in order to obtain the total costs per design.

4.3.3 Service level violations

The second component of the total costs is formed by penalty costs that are invoked if response times suffer from resource bottlenecks and exceed given SLAs. This can be a result of too aggressive consolidation attempts as described in Section 3.3.4. These costs are to be identified on the basis of the prediction results. For the sake of demonstration, a sample SLA is designed, as also used by Gmach et al. (2008). According to the SLA, 99% of the requests to change sales orders must be processed within a deadline d_1 of one second (also referred to as deadline constraint d_1). For each percent point under fulfillment, penalty costs of 150$ are invoked using a continuous scale. As defined in Section 3.3.4, requests may be aggregated to hours of (non-)fulfillment. In Equation 4.2, n represents the total number of considered hours and v represents the number of hours in which the SLA is violated.

$$c_{sla} = \begin{cases} \frac{v*100}{n} * 150 & \text{if } \frac{v*100}{n} \geq 1 \\ 0 & \text{if } \frac{v*100}{n} \leq 1 \end{cases} \tag{4.2}$$

Optionally, an additional deadline constraint d_2 defines a threshold of 1,500 milliseconds. Accordingly, designs which induce response times higher than d_2 are unacceptable even if they minimize the total costs, e.g., due to user satisfaction. In the described use case, the total number of considered hours n accounts to 587 in the measured time frame. On the basis of the predicted mean response times for each hour, load scenario, and solution candidate, the 18 future scenarios are associated with SLA violation degrees. According to Equation 4.2, resulting penalty costs are calculated as presented in Table 4.6. The outcome may serve as decision support for capacity management already in the presented form. For example, if deadline constraint d_2 is strictly active and load factors up to 2 are conceivable in the near future, Design A is to be excluded from the solution space, since

Design	Load factor	Violated hours (d_1)	Violated hours (d_2)	Fulfillment degree	Violation degree	Penalty costs
A	1	4	0	99.32	0.68	0
A	1.25	9	0	98.47	1.53	229.5
A	1.5	35	0	94.04	5.96	894
A	1.75	153	0	73.94	26.06	3,909
A	2	337	1	42.59	57.41	8,611.5
A	2.25	439	2	25.21	74.79	11,218.5
B	1	0	0	100	0	0
B	1.25	1	0	99.83	0.17	0
B	1.5	17	0	97.1	2.9	435
B	1.75	138	0	76.49	23.51	3,526.5
B	2	276	0	52.98	47.02	7,053
B	2.25	366	3	37.65	62.35	9,352.5
C	1	0	0	100	0	0
C	1.25	0	0	100	0	0
C	1.5	6	0	98.98	1.02	153
C	1.75	50	0	91.48	8.52	1,278
C	2	189	0	67.8	32.2	4,830
C	2.25	355	0	39.52	60.48	9,072

Table 4.6: Penalty costs for the 18 future scenarios.

violations are to be expected. If any load scenario may appear, the deadline d_2 allows for the deployment of Design C only. In Table 4.6, the future scenarios which violate d_2 are grayed out. If, on the other hand, d_2 was not defined, all future scenarios are to be analyzed with respect to the their total costs. In this regard, the assumed relation between operations costs and SLA violation probability is confirmed by the plot in Figure 4.17. Here, the six lines are formed by three points, each representing the three designs A, B, and C. The y-axis refers to the number of violated hours with respect to the deadline d_1. Especially in a scenario of doubled load, Design C is able to reduce the number of violation hours significantly when compared to Design A and B. Any workload above this factor is hardly manageable for any of the Designs. In contrast, lower load factors around the current workload may also be handled by designs which highly utilize a minimized amount of capacity such as Design A. In general, additional capacity reduces the risk of SLA violations but increases operations costs. Therefore, cost minimization is subject to the subsequent total cost analysis.

4.3.4 Total cost analysis

In the preceding steps, operations costs and penalty costs were computed for each future scenario. Furthermore, designs which do not fulfill the deadline constraint d_2 of a given SLA may have been excluded. Therefore, the final activity of the future scenario analysis is to identify a design that minimizes the total costs across the remaining solution space. First, total costs are to be computed for each future scenario on the basis of Equation

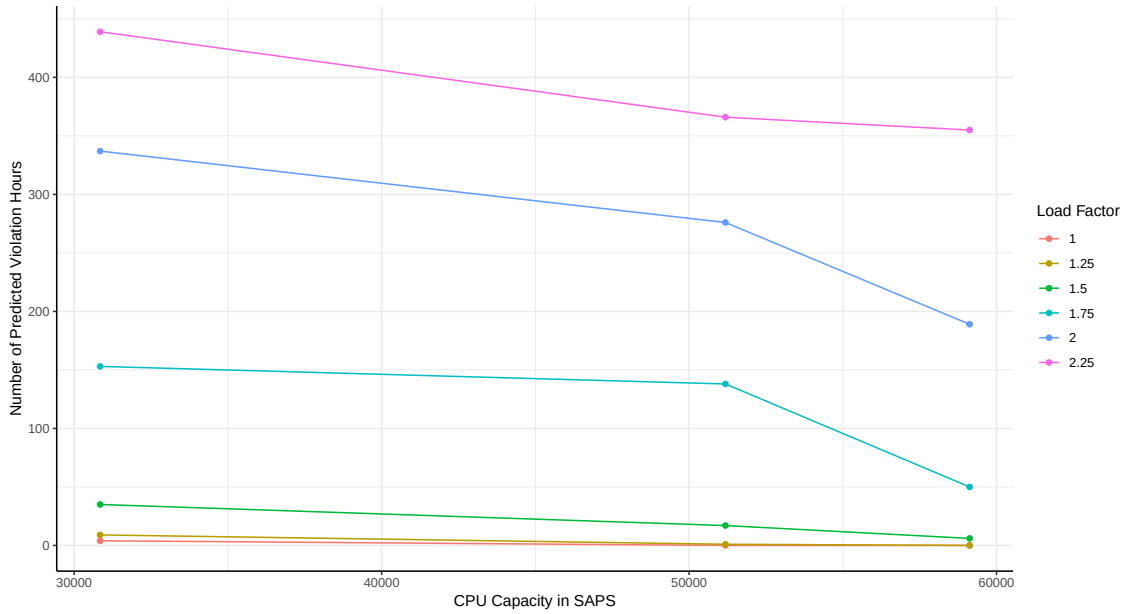

Figure 4.17: Capacity requirements and hours of SLA violation for each design.

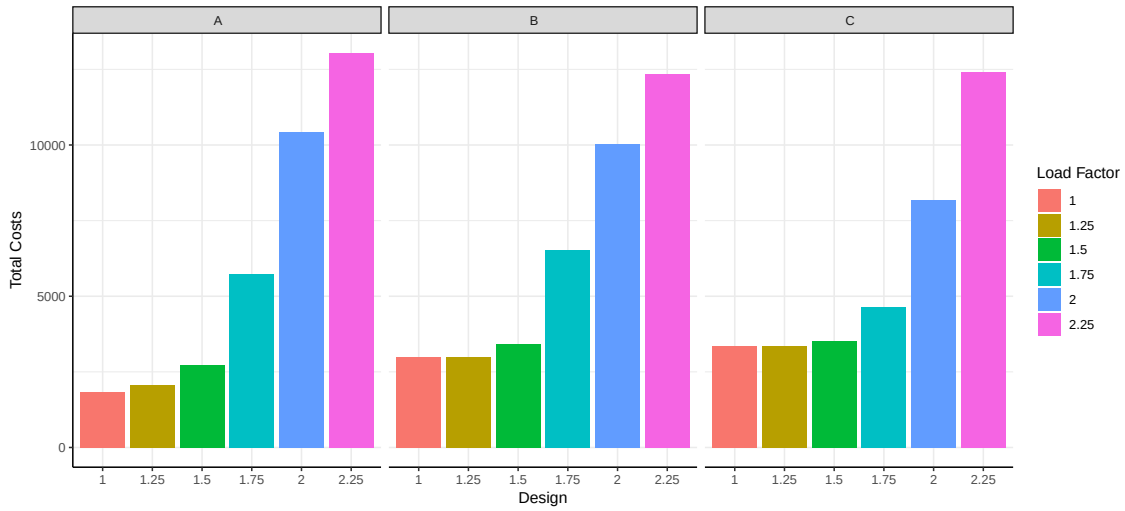

Figure 4.18: Total costs for each future scnenario.

3.7. Exemplary results for the studied case are illustrated in Figure 4.18. As stated in Section 3.3.5, the illustration of total costs is part of the capacity plan executive summary. Depending on the deadline d_2, however, certain designs may be grayed out. The final design recommendation depends on the expected workload. If, for example, the current level of workload was also forecasted for the near future, Design A would sufficiently minimize the total costs. In case of mainly doubled load, Design C performs most cost-effectively. In order to resolve the given dependencies, probabilities may be obtained for each load scenario on the basis of the business strategy. Respective input describes how often each load extend is expected to occur and, in turn, enables to weight the total costs per design. In order to demonstrate design recommendations, three alternative story lines are considered for the studied case of a large retail company:

- **Storyline 1:** The capacity manager expects business as usual. This storyline covers an alternative option to the introduced centralization project and gets activated in case the planned changes are canceled.
- **Storyline 2:** About 125% of additional markets are to be integrated into the existing system landscape in the near future. Significantly increased load is expected to dominate the EA usage.
- **Storyline 3:** The capacity manager expects mainly business as usual. However, few additional load peaks may result from the integration of a small number of test markets before major changes are planned.

The story lines exemplify expected changes that are to be considered when managing IT capacity. Therefore, each storyline leads to different decision support as part of total costs minimization. Resulting design recommendations are given in Table 4.7 for the studied case. Consequently, the derivation of load probabilities from alternative story lines allows to provide a single recommendation. In these cases, the decision support process can be automated by minimizing the weighted total costs. However, as described in Section 3.3.5, recommendations must be justified and underlying assumptions must be described in the capacity plan. Therefore, presented illustrations and discussed dependencies (cf. Section 4.3.3) represent input to the capacity plan.

Load factor	**Probability (storyline 1)**	**Probability (storyline 2)**	**Probability (storyline 3)**
1	1	0.1	0.8
1.25	0	0	0
1.5	0	0	0
1.75	0	0	0
2	0	0	0.2
2.25	0	0.9	0
Recommendation:	Design A	Design C	Design B

Table 4.7: Design recommendations on the basis of load probabilities.

To conclude, the demonstrated future scenario analysis effectively utilizes performance models in order to address the secondary objective of the PPSS method. The obtained estimations of SLA violations enable to identify solution candidates which minimize the total costs. Hence, the utility of a machine learning-based prediction technique which uses shared performance counters could be demonstrated.

4.3.5 Summary of EVAL 3.2

Utility was demonstrated scenario-based using a case study with real data. Since capacity management becomes challenging and particularly relevant whenever changes are planned, a large retail company was introduced to face IT integration plans. As part of the case study, three solutions to the SPP were computed, representing alternative designs that are to be evaluated with respect to their violations of performance SLAs and their total costs. Therefore, solution candidates serve as input to the step *Predict service performance*. The subsequent *Future scenario analysis* confirmed that greater amount of capacity generally reduces response times and their variance in the investigated case. On the basis of the predicted values, a sample SLA was applied and resulting violation degrees could be computed for alternative load scenarios. As a result, 18 future scenarios (combination of load scenario and solution candidate) were associated with penalty costs. Operations and penalty costs form total costs. Depending on the probability of expected load factors, total costs could be minimized and design recommendations were derived. Hence, performance models which were trained on a cross-organizational basis could effectively be utilized to evaluate solution candidates to the SPP regarding selected performance characteristics.

4.4 Technical implementation and field usage (EVAL 4)

This section evaluates the application of the artifact in the environment (cf. Figure 1.1 on the left). Relevant evaluation criteria of EVAL 4 are the ease of use, the operationality, the fidelity with real world phenomena, as well as the impact on the artifact environment and user. Therefore, in a first step, the technical implementation of the artifact is described. This way, a prototype of the artifact could be offered to a group of pilot users who were then surveyed to provide feedback from the productive usage in the field. The feedback is limited to the implemented features. Hence, it must be noted that some of the evaluation criteria refer to individual components of the artifact, so that EVAL 4 is carried out only partly. The described technical implementation serves as a sample setup to proof the above mentioned criteria, however, the selection of appropriate software and hardware is up to the capacity management provider as the designed artifact is independent of a particular technology.

4.4.1 System architecture and workflow

The PPSS method relies technically on a central knowledge base. Three layers of this APM-KB were introduced in Section 3.1.3. While the data layer integrates APM data

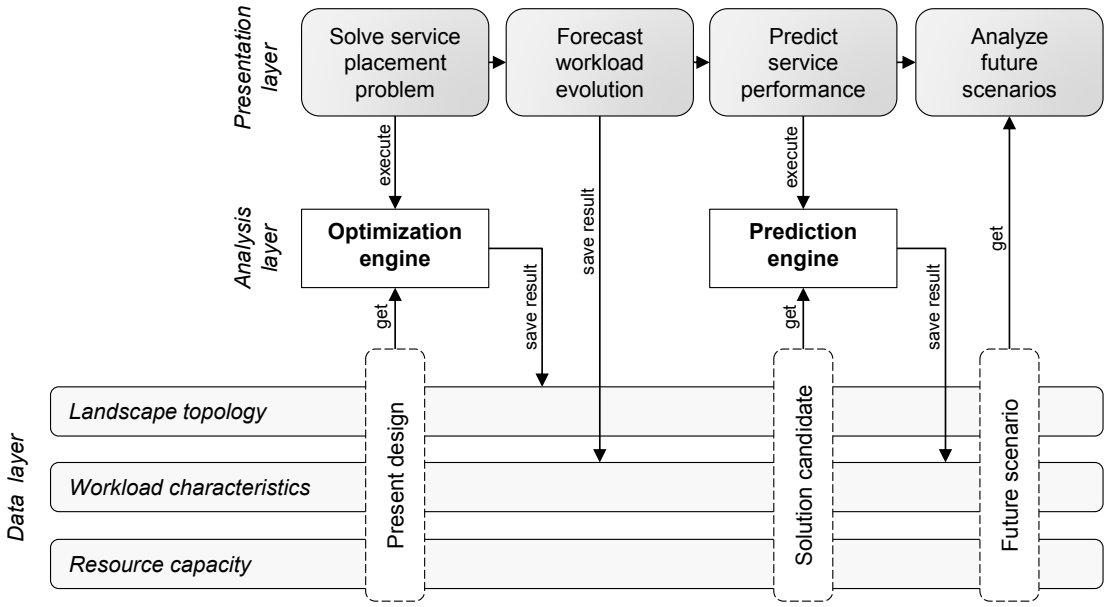

Figure 4.19: Layers of the APM knowledge base and their interplay.

from participating environments, the analysis layer offers computational intelligence, that is optimization algorithms and prediction models. Finally, on presentation layer, a web interface supports the tasks of the designed process. Figure 4.19 illustrates the interconnection of these layers using an excerpt of the workflow from a capacity management consumer perspective. On the data layer, a relational database schema is created which holds all attributes relevant to optimization and performance prediction. This includes meta data regarding the landscape topology, workload characteristics, and resource capacity limits. Workload characteristics are grouped by hour and associated with a time stamp. Data is generally aggregated to the layers, presented in Figure 2.1. Topology tables link these layers to each other, i.e., providing which service is running on which server. For each server, capacity limits in terms of CPU and memory are present. All data is grouped by a measurement ID which identifies a monitoring season of a data center. The ID also enables role-based access control. In the instantiated artifact, the data layer is implemented using the in-memory database platform SAP HANA. At this point, all monitoring data is loaded into main memory in order to support a service delivery in accordance with functional requirement F9 and quality requirement Q3. However, least accessed measurement IDs may be unloaded for the sake of scalability. An identical spare server ensures to fail-over the complete APM-KB in case of hardware issues. The concept is enabled by the underlying application virtualization technique (cf. 2.2.2).

The optimization engine holds the heuristics, metaheuristics, and hybrid algorithms and, if executed by the user, solves the SPP. For this purpose, the data layer is queried for the present design, including relevant data which forms the workload profiles and the server capacity limits. Solution candidates represent alternative topologies and, therefore, are stored in the topology-related tables. Results of the subsequent step *Forecast workload*

evolution (cf. Presentation layer in Figure 4.19), represent alternative workload profiles and are stored in the workload tables respectively. The combination of workload profiles, optimized topology, and associated capacity limits is passed to the prediction engine which estimates the resulting performance. Prediction results, i.e., mean response times, are saved together with workload characteristics in the respective tables. A solution candidate that was supplemented by a load scenario and resulting response times forms a possible future scenario. These scenarios are subject to further analysis, e.g., with respect to their weighted total costs, in the last step of the process.

The presentation layer is technically implemented on the basis of an integrated web server which is part of the SAP HANA platform. Therefore, the graphical user interface was implemented using the software development toolkit (SDK) SAPUI5, which provides libraries on the basis of HTML5 and supports user interface development according to the Model-view-controller pattern (Leff and Rayfield, 2001). However, alternative SDKs are just as conceivable for the purpose of data visualization. Data is retrieved from the data layer using JavaScript. The subsequent section describes the analysis layer in more detail.

4.4.2 Back end engines

The analysis layer holds two back end components: The optimization engine and the prediction engine.

The optimization engine is implemented as a JAVA servlet which provides optimization as a web service. This way, the service is available over the internet and can be integrated into the APM-KB user interface or directly invoked using the respective URL. The algorithms are initiated by passing workload profiles and server capacity limits in the JSON format. Output, in the same format, is written to a database table where it may be processed further, e.g., by the prediction engine. Although the user interface allows to select the appropriate algorithm type for a particular problem, the GGA is, according to the results of EVAL 3.1, the default choice. The servlet is deployed on a separate virtual machine which runs the open-source web server Apache tomcat. The reference architecture for the algorithm implementation is described in Akhras (2017).

The prediction engine, on the other hand, is implemented by means of a server which hosts the server component of the software environment R for statistical computing. The R server was integrated with the SAP HANA database as described in SAP (2018b). Accordingly, database stored procedures of the language R were implemented to query the input data, pass it to the R server, and store the prediction result in an output table. As part of the process, the actual R functions are executed on the separate R server. This way, resource intensive operations like model training do not interfere with daily operations of the APM-KB. In order to maintain the performance models and to track hyperparameters and error metrics, the data model, depicted in the UML class diagram of Figure 4.20, was created. Using the table *MODELPARAM*, hyperparameters for model training can be configured as a key-value-store. The ID *PARAMETERSET* refers to a particular setting. In the table *MODEL*, where accuracy results are stored

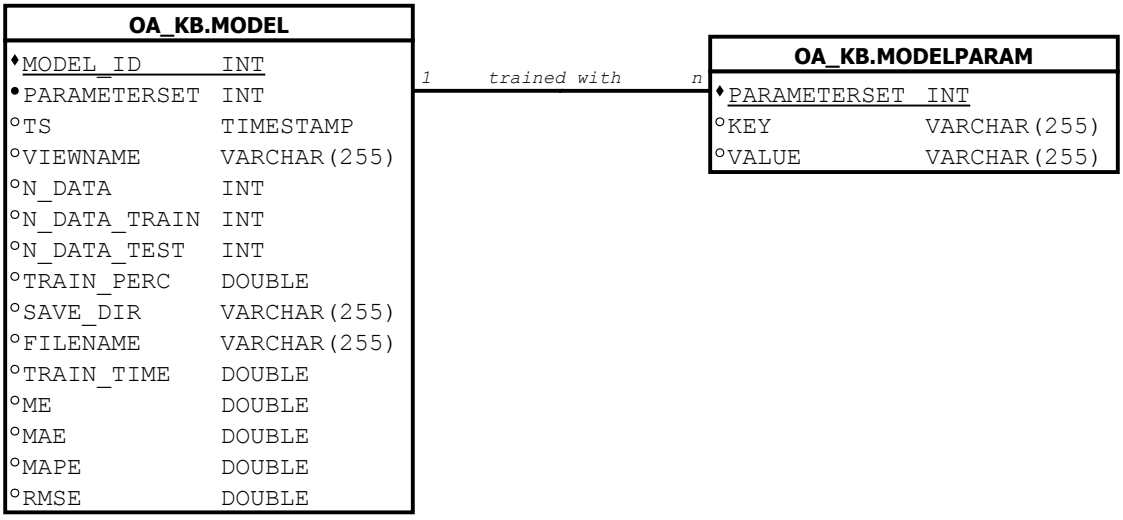

Figure 4.20: Data model for performance model meta data.

for each model, the *PARAMETERSET* is used as a foreign key to associate models with their hyperparameters. Trained models are stored in the file system of the R server and the respective destination is stored in the table *MODEL* too. In order to construct and apply models, the stored procedures *train* and *predict* were implemented. When executing *train*, a database view which holds the training data and a parameter set must be passed. Training data is split by the procedure to allow for cross-validation. After model training, the procedure stores all results in the table *MODEL*. Similarly, when executing *predict*, a database view is to be passed, holding all input records for which the response time is to be estimated. On the basis of the provided database view, the procedure queries the table *MODEL* for a suitable model which provides the lowest mean absolute percent error. Prediction results are typically computed within seconds and stored in an output table of the same schema. This way, the future scenario analysis is enabled.

4.4.3 User interface

A graphical user interface (GUI) was developed to interact with the optimization engine as part of the task *Solve service placement problem*. Furthermore, the tasks *Understand the environment* and *Characterize workloads* are supported by the GUI, as these are main steps to precede any kind of further analysis. A landing page, to which any user of the APM-KB is forwarded, allows to navigate to a particular interface which supports one of the PPSS tasks.

As part of the initial descriptive tasks, the GUI allows to analyze a number of metrics from both services and server perspective, using line charts, bar charts, and stacked bar charts. The selectable metrics correspond to the attributes, listed in Table 3.3. Observations may be filtered on the basis of the time stamp and, if desired, grouped by weekday, day, or hour of the day. In Appendix C, screenshots of the user interface exemplify the

implementation. The example in Figure C.4 allows to identify peak usage periods which, according to Almeida (2002), is an important step when understanding the environment.

If, e.g., the heat map in figure C.5 reveals low peak utilization levels, the GUI allows to instantiate an SPP by retrieving server capacity limits and workload profiles for all running services in an automated way. As the optimization engine allows to model individual constraints, an interface component was designed to support this process. As shown in Figure C.6, a drop down menu allows to select constraint types. Depending on the selection, servers or services can be chosen to which the constraint applies. As described in the conceptual design (cf. Section 3.1), a feature was implemented to save modeled sets of constraints in the APM-KB for future reuse. If appropriate, an algorithm which differs from the default GGA may be chosen and results of the optimization run are presented to the user by means of a simple allocation table and a bar chart which visualizes resulting server utilization levels. The individual characteristics of the SPP may be refined in multiple iterations until a number of feasible solution candidates were computed.

The prediction engine, in contrast, is currently invoked over an SQL interface. Therefore, the tasks which intend to predict service performance and to analyze future scenarios are carried out on a command-line level and, in this form, were not offered to the pilot user group. The implementation of a suitable GUI is planned as future research and development. In this regard, the landing page, shown in Figure C.1, indicates some interfaces of the APM-KB that are not directly related to the PPSS method. These interfaces result from additional research streams in which the extendability of the APM-KB with respect to further types of data analysis was explored. Respective research streams, however, are out of the scope of this thesis.

4.4.4 User feedback

User feedback helps to evaluate characteristics of the artifact which become relevant when being applied in its intended environment. Consequently, the presented GUI was offered to two data centers whose employees participated in a pilot phase of six weeks. As the artifact is planned to be offered by a commercial capacity management provider, this phase was also part of the go-to-market strategy. From a research perspective, the pilot phase aimed at two main objectives:

- Evaluation of the PPSS components which are currently available through the graphical user interface on the presentation layer of the APM-KB. In particular, the pilot phase seeks to evaluate the operationality of the method, the efficiency of the method with respect to cost savings, the ease of use, and the impact on the artifact environment.

- Prioritization of future research and development efforts. The pilot phase is expected to reveal possible improvements and missing features, thus, contributing to the evaluation of completeness.

	Data center A	**Data center B**
Company industry	retail	insurance
Company category	large	large
Number of services	69	20
Number of servers	53	7

Table 4.8: Profiles of the two participating data centers.

To enable the individual instantiation of the service placement problem, an initial measurement phase has preceded the actual pilot phase. During that, all running services and servers were monitored at each data center for a period of three weeks. Measurement data was collected and imported into the data layer of the APM-KB. Subsequently, pilot users were provided with accounts which allow to authenticate at the APM-KB and to solve individual instantiations of the SPP. Furthermore, descriptive analyses were offered to support the tasks *Understand the environment* and *Characterize workload* (cf. Section 4.4.3). All features were offered free of charge but the intention to charge prices in the future was communicated. Details on the two participating data centers are provided in Table 4.8. User feedback was collected two-fold. On the one hand, an online survey was available from within the service (cf. Figure C.1) throughout the whole pilot phase. On the other hand, a face-to-face expert interview was hold with each data center individually in order to obtain unbiased feedback. Here, five users from data center A and three users from data center B participated. The questionnaire of the online survey, at the same time, served as an agenda which guided through the meeting. This way, participants were able to explain their given answers in more detail. Answers to the questionnaire were grouped by data center and, in this form, are highlighted in Table 4.9. Here, the term landscape optimizer refers to the GUI component which executes the optimization engine when supporting the process step *Solve service placement problem.* The answers, presented in Table 4.9, are discussed in the following.

The first two questions address the research goal that is to develop a method which reduces costs in capacity management. In this regards, both data centers are convinced to save time and to reduce manual effort, compared to their current approach (cf. Question 7). Naturally, savings affect exercises which were simply outsourced to the capacity management provider. However, algorithms and models, once constructed by the provider, are designed to be offered to a great number of consumers within the domain of a COTS EA. Therefore, economies of scale are expected to reduce overall costs too.

Question 3 and 9 refer to placement constraints, contributing to the artifact's fidelity with real world phenomena. First, the general usefulness was questioned. Answers were analyzed in Section 2.2.5 and Section 3.2.3 and, therefore, are not discussed in further detail. As stated in Section 3.2.3, the constraint type, requested by one of the data centers, results from given downtime restrictions during the deployment of a solution candidate. To summarize, particular services must be available if others go down. These dependencies could be addressed by limiting the initiated SPP to a subset of services or

1. Compared to your current capacity management approach, how does the offered service affect the overall amount of required time to get to desired answers and solutions to capacity management challenges?

more time				no effect					less time
1	2	3	4	5	6	7	8	**9**	10

2. Compared to your current capacity management approach, how does the offered service affect the overall amount of manual effort to manage your infrastructure capacity?

more effort				no effect					less effort
1	2	3	4	5	6	7	8	**9**	10

3. How useful do you rate the possibiltiy of defining placement constraints for the optimization task?

not useful									essential
1	2	3	4	5	6	7	8	**9**	**10**

4. Do you consider the result of the landscape optimizer as realistic in terms of expected quality of service, e.g., with respect to targeted service response times?

unrealistic				not sure					realistic
1	2	3	4	**5**	6	7	**8**	9	10

5. How frequently would you use the final version of the landscape optimizer in the future?

never				yearly					monthly
1	2	3	4	**5**	6	7	8	9	10

6. How frequently would you use the descriptive analysis features which support to understand the environment?

never				yearly					monthly
1	2	3	4	5	6	**7**	**8**	9	10

7. How do you currently optimize your landscape with respect to server utilization?
   A. Automated, by means of optimization algorithms.
   B. Manually, by estimating service demands and mapping them to server capacities.
   C. We do not relocate existing services in order to optimize server utilization.
   D. Other.

8. How frequently do you optimize your landscape?
   A. In real-time.
   B. Approximately six-monthly.
   C. Approximately yearly.
   D. Rarer then yearly.
   E. Never.

9. Is there a placement restriction that you cannot model using the provided constraint types? Which placement constraint is missing?
   Partial optimization.

10. Which additional functionality are you missing?
    Workload trend analysis; Reporting templates; Direct connection to the database.

Table 4.9: Aggregated user feedback as part of the online survey.

servers, thus, performing a partial optimization. The request was addressed by the design and implementation of a *Component exclusion constraint* which is also available in the current GUI version (cf. Figure C.6).

Question 4 intends to obtain an impression with respect to trust. As the prediction engine was not provided to the pilot users, the questioned result of the landscape optimizer refers to solution candidates which did not undergo a performance evaluation. The answers indicate that means to improve the credibility of results, e.g., predictions of resulting response times, will most likely be appreciated by the users.

Answers to questions 5 and 6 emphasize the utility and the relevance of the artifact as both data centers intend to purchase the offered services in the future on a regular basis. In both data centers, capacity is currently managed with significant manual effort. Resource demands are estimated and mapped to servers of appropriate capacity manually. Alike procedures are carried out approximately every year respectively every six months in order to optimize the current landscape.

Finally, participants were asked to request additional functionality in order to guide future research and development directions. With respect to the descriptive analysis, both data centers requested features which bring the presented results in a format that is easily interpretable by the management. This way, participants expect to simplify reporting and decision preparation. On the other hand, a direct SQL connection to the data layer of the APM-KB was requested to allow for additional in-depth performance analysis by application and database administrators. One of the data centers, furthermore, requested a feature to analyze workload trends over a longer period of time. This way, the currently manual task *Forecast workload evolution* would be supported by an additional component on the analyses layer of the APM-KB.

During the expert interviews, all of the participants stated that the developed artifact allows to manage the capacity with increased efficiency when compared to the current approach. According to the users, the provided analyses bridge the existing gap between application and infrastructure teams as capacity bottlenecks for particular business tasks are getting revealed. A user of data center A stated that, in the current practice, capacity is increased in situations of performance bottlenecks. However, if idling resources are identified, a lack of courage prevents from decreasing capacity in the same manner. The user expects the data-driven approach to increase acceptance of capacity decreasing decisions. With respect to the ease of use, intuitive usability was generally emphasized although minor layout and color changes were suggested. For example, the color scheme to visualize server utilization may follow traffic light colors to quickly indicate under- and overloads.

4.4.5 Summary of EVAL 4

Evaluation on level 4 involves to put the artifact into use. Central backbone of the PPSS is a knowledge base which holds APM data. In order to evaluate operationality, the three layers of the APM-KB were implemented by means of a database system which integrates a

web server for the presentation layer. The analysis layer is served by an optimization engine and a prediction engine. While the former was implemented as a JAVA servlet, the latter utilizes an integrated R server. Results of the optimization engine are written to tables on the data layer of the APM-KB. These are passed to the prediction engine for performance evaluation of solution candidates. Meta data of performance models, including parameter sets, are stored in the APM-KB too. This way, model selection was automated on the basis of input data (type of business transaction) and accuracy metrics. The tasks *Solve service placement problem*, *Understand the environment* and *Characterize workloads* of the PPSS are supported by a graphical user interface which allows to interact with the APM-KB on the presentation layer. The GUI was offered to a group of pilot users from two different data centers who took the role of capacity management consumers with the objective to evaluate the ease of use, the fidelity with real world phenomena and the impact on the artifact environment. User feedback was collected by means of an online survey and two semi-structured interviews. Results indicate cost savings for the participating data centers in the supported capacity management exercises. Results further confirm the necessity to support placement constraints and solution performance prediction. Finally, the ease of use was emphasized and led to the will to consume the offered service on a half-yearly to yearly basis.

4.5 Summary of the artifact's evaluation

Evaluation on the levels EVAL 1 and EVAL 2 concerns the relevance, importance, novelty, and the general applicability of the research artifact. Therefore, chapter 2 outlines the state of the art in the domain of capacity management along with existing challenges with respect to the data center efficiency. Applicability of the research artifact was theoretically analyzed on the basis of a usage analysis of COTS EAs. As a result, standardization degrees are sufficiently high in order to follow the targeted strategy of cross-organizational learning from shared performance counters. Practical applicability was evaluated as part of the design phase by means of a data mining process which led to performance models for predicting mean response times of business transactions. As these were trained and cross-validated on the basis of APM data from different environments, applicability of the approach was further confirmed. Here, machine learning models which were trained using Random forests and Boosted trees revealed acceptable results that meet the quality requirement Q2. Having confidence about the applicability, the designed artifact was evaluated against the requirement specification (cf. Table 3.1) and the evaluation criteria (cf. Table 1.1) in Chapter 4. For this purpose, two evaluation methods were applied on the level of EVAL 3: First, an experimental field study was conducted to evaluate the service placement approach with respect to solution quality and consistency as well as algorithm efficiency. Second, a case study demonstrated the utility of the performance prediction approach using the previously validated models.

The field experiments utilized APM data from 516 real data centers of which each was monitored for a period up to three weeks. A descriptive analysis of the data re-

vealed a large variety of the environments under study with respect to, e.g., the server sizes, the number of running services, and the extent of workloads. Hence, generality of the approach could be confirmed within the boarders of the defined application area. The designed service placement approach involved eight different algorithm types to solve problem instantiations. These were tested in three different scenarios for each of the 516 data centers. The scenarios involve varying extents of constraint pressure. The setup resulted in a total number of 12,384 experiments which allowed to analyze solution quality in relation to the algorithm type and the problem class. A solution represents an optimized design which imposes a certain allocation of services to servers. Solution quality was defined by a fitness value which considers the amount of capacity savings. To foster high solution applicability, the fitness values was penalized by resource overflows and constraint violations. Since the problem is known to be NP-hard, optimal solutions could not be computed as a reference value. Instead, solution fitness was compared to best values across all algorithms within a problem class. Therefore, the mean distance to best fitness served as one of the evaluation metrics. However, relative savings and constraint violations were also analyzed in a separate step in order to investigate the strengths and weaknesses of the algorithms in more detail. For evaluation of the economic relevance, mean server utilization levels for the original and the optimized designs were compared. Finally, algorithm run-time was provided. With respect to the fitness difference, heuristics provided high-quality solutions in unconstrained scenarios. A pure genetic algorithm identified best solutions in constrained scenarios and worst solutions in unconstrained scenarios. Interestingly, the grouping genetic algorithm outperforms the GA by far in the unconstrained scenarios with solutions whose quality is on the level of the heuristics. Best constraint compliance was achieved by the GA, followed by the GGA which seems to suffer slightly from repair mechanisms to ensure solution feasibility after recombination. In terms of capacity savings, the two metaheuristics perform equally well. In unconstrained scenarios, solutions require in average only 44% of the original server capacity. Across all scenarios, mean savings of at least 22% were achievable. In general, saving potential increases with the complexity of a case. If the problem class is unknown at the time of algorithm selection, the GGA provides suitable solutions with highest probability. Run-time accounted to less then 2.3 seconds in every experiment and, therefore, does not limit the set of applicable algorithms in the domain of offline consolidation. Regarding the requirement specification, the compliance with the functional requirements F1 and F3-F7 as well as quality requirement Q4 could be confirmed by solving respective instantiations of the SPP in a variety of cases. In turn, quality requirements Q1 and Q3 are met by the analyzed solution quality and the algorithm run-time.

The second part of EVAL 3 addressed the evaluation of the PPSS tasks to predict performance and to analyze future scenarios. Therefore, a case was designed on the basis of real data and the SPP was initiated for this case with three different maximum utilization constraints. Accordingly, three solution candidates represent optimized designs of different risk attitudes. The workload and capacity values associated to these solution candidates

formed input to the performance model. The prediction results, in turn, provide the basis for the future scenario analysis. Results confirm that response times decrease with increasing capacity. Required servers could be associated with operations costs depending on their size. Using a sample SLA, predicted response times were used to calculate SLA violation degrees under certain load situations. On this basis, expected SLA penalty costs were computed and added to the operations costs in order to obtain the total costs for each future scenario. Finally, it was demonstrated that, depending on the expected future load, alternative designs must be recommended to minimize the total costs. Therefore, functional requirements F1, F2 and F8 are met. In the designed case study, the PPSS helped to identify cost-effective design alternatives and, in conclusion, was of utility for the artifact's environment.

Finally, it was the goal of EVAL 4 to implement the artifact for productive usage and observe its operationality and its impact on a real environment. Accordingly, the central component of the PPSS, the APM knowledge base, was implemented on all three layers, utilizing an in-memory database system, an R server (as prediction engine), and a JAVA servlet that was deployed on a Tomcat web server (as optimization engine). On the presentation layer, a GUI was implemented to offer functions of the PPSS in accordance with functional requirement F9. These were offered to a group of test users from two different data centers. By means of the resulting feedback, further criteria on the level of EVAL 4 could be evaluated. In particular, ease of use was confirmed by both organizations. As the users initiated problem instances of the SPP on the basis of their individual APM data, the fidelity with real world phenomena is provided. Furthermore, the confirmed reduction of time and effort, compared to their current way of managing capacity, indicates an impact on the artifact environment which allows to save costs.

5 Conclusion

Research involves to communicate both the benefits and the drawbacks of its results. It is the goal if this final chapter to summarize the work and its scientific contributions. At the same time, existing limitations are discussed. These serve as baselines for future research.

5.1 Summary of the work

The thesis explores a method for enterprise application consolidation exercises with the objective to reduce costs and to improve solution applicability. According to the formulated hypothesis, EA usage profiles are dominated by standard functionality. Therefore, the potential to reduce costs resides in increase of standardization and decrease of expert knowledge. The strategy to address this potential envisages to leverage black-box techniques with existing monitoring data. The thesis follows the design science methodology and applies the extended design-evaluate-construct-evaluate pattern according to which evaluation is carried out on four levels throughout the work.

As the concept relies on the hypothesis, first, the customization degree of a COTS EA was analyzed, contributing to the evaluation of relevance and feasibility on level one. Results indicate a mean customization degree below 20.12% on the basis of the invoked number of transaction calls. Furthermore, large environments tend to show higher standardization as opposed to small and medium environments. This finding complements well with the optimization potential of server consolidation efforts which also improves with an increasing number of running services and servers. In the targeted domain, mean server utilization levels were analyzed to be around 8%. Consequently, server consolidation was introduced to be an effective tool to reduce idling capacity and improve efficiency. The problem characteristics correspond to a combinatorial optimization problem which is known to be NP-hard. Related problems are typically solved by means of heuristics and metaheuristics. However, solution strategies vary in terms of solution applicability. None of the studied approaches supports heterogeneous servers, multiple resource dimensions, dynamic workload profiles and placement constraints at the same time. Furthermore, existing SLAs which are related to response times were considered only indirectly. This research gap was addressed by the thesis at hand. In accordance with the hypothesis, the outlined strategy to achieve the research goal was followed when specifying requirements for the research artifact. The designed research artifact represents a method for managing EA capacity.

The method addresses the research goal by outsourcing capacity management tasks

which typically require costly expertise to a capacity management provider. This approach has multiple benefits: The provider follows a structured data mining process. Therefore, performance models, once evaluated, can be applied by a large number of capacity management consumers within the domain of the EA vendor, leveraging economies of scale. Second, central retention of monitoring data increases the variety of training data with respect to alternative workload and hardware characteristics for all participants. Capacity management consumers are enabled to simulate change effects on the basis of real observations from other environments with well-known accuracy metrics. These changes may result, e.g., from server consolidation activities whose effect on the transaction performance is worth knowing in advance. Finally, the consumers can concentrate on the core competencies of their daily operations without the need to gain knowledge in the domains of machine learning and combinatorial optimization.

The method was modeled using the business process modeling notation. The essential tasks of the process are *Solve service placement problem*, *Predict service performance*, and *Analyze future scenarios*. Required constructs to manage these tasks were designed form a capacity management provider perspective. Hence, optimization algorithms and performance models were built. The former address an offline optimization problem which aims to identify design alternatives that reduce operations costs. This service placement problem was formulated in a way which allows to consider dynamic workload profiles of the running services in multiple resource dimensions. Furthermore, heterogeneous servers with different capacity limits and ten placement constraint types are supported. These types result from both the scientific literature and from expert interviews with practitioners. To solve individual problem instantiations, four heuristics, two metaheuristics, and two hybrid algorithms were selected. While the heuristics are suitable for unconstrained scenarios, metaheuristics are designed to evaluate placement constraint compliance as part of their fitness function. The hybrid algorithms use a metaheuristic which optimizes the input sequence to the placement heuristic. As part of the designed method, optimization algorithms compute solution candidates which are subject to subsequent performance evaluation.

For this purpose, the Cross-industry standard process for data mining, as incorporated into the PPSS method, was gone through. Using the example of the SAP business transaction to change sales orders, performance models were constructed and evaluated on the basis of real monitoring data. Bagging (Random forest) and boosting (AdaBoost) techniques revealed accuracy values which are considered to be acceptable in the domain of capacity management. The results contribute to the evaluation of applicability on level two. Across the ten most frequently used transaction types within the sales and distribution module, the boosting algorithm AdaBoost, in conjunction with regression trees, achieved constantly best model accuracy with a mean absolute percent error between 19.57% and 29.34%. Evaluated performance models may be applied by the capacity management consumer to predict any change effects. However, PPSS suggests to use the models for evaluating the performance of solution candidates to the SPP in order to allow

for the minimization of SLA violations. Total costs of a solution candidate are defined as the sum of their operations costs and their SLA penalty costs. The future scenario analysis aims at the minimization of total costs within a possible future scenario. Depending on how well future workloads are known a priori, this step leads into a single design recommendation or recommendations for each load scenario. Both variants serve as decision support in order to minimize costs.

Evaluation on level three was carried out by applying the designed constructs from a capacity management consumer perspective. Therefore, the eight algorithms were used to solve realistic optimization problems on the basis of measurement data from 516 real data centers. To test algorithm suitability for different requirements, one unconstrained scenario and two constrained scenarios were built for each of the data centers, resulting in a total number of 12,384 field experiments. In the constrained scenarios, a set of four realistic placement constraints was defined where a maximum utilization constraint limits server utilization levels to an upper bound of 85% respectively 65%. Solution applicability is determined by the compliance with given realities and constraints. Therefore, solution quality is defined by the amount of resource savings which is penalized by resource overflows and constraint violations. With respect to these targets, a grouping genetic algorithm performed well across all experiments. While a pure genetic algorithm failed to compute solutions of competitive quality in unconstrained scenarios, it performed best with increasing constraint pressure and surpassed the solution quality of the GGA. Heuristics solved unconstrained problem instantiations with sufficient solution quality in shortest time (below 0.3 ms). However, mean execution time accounted to less then 2.3 seconds for all experiments and, in this magnitude, is not an issue in offline consolidation approaches. In unconstrained scenarios, average resource savings accounted to 56% of the original capacity. Across any scenario, mean savings of at least 22% were achievable.

To evaluate the performance prediction task, a case study was designed. Three solution candidates, computed by the GGA, represent design alternatives of different capacity requirements and risk attitudes. Using the evaluated performance model, mean response times per dialog step could be predicted for the medium-complex standard business transaction to change sales orders. Predictions were made for both the historical workload and alternative future workloads which result from the business strategy. Results were investigated as part of the future scenario analysis. According to the results, mean response times and their variance reduce with increasing capacity. Hence, solution candidates of higher total capacity provide more stable designs with better performance. To enable design recommendations, prediction results were used to determine violation degrees with respect to a sample SLA. According to this SLA, penalty costs were computed for each solution candidate under each workload factor. Furthermore, capacity requirements were translated into operations costs. This way, designs which minimize the sum of penalty costs and operations costs, were identified. In order to demonstrate total cost minimization in an a priori approach, sample probabilities for certain load factors were aggregated to three story lines. On the basis of this input, the total cost minimization for each story

line led to a different solution recommendation.

Evaluation on level four required to put the artifact into use. To evaluate operationality, the three layers of the APM knowledge base were implemented using an in-memory database management system. The data layer integrates measurement data from more than 6 billion transaction calls, invoked on over 18.000 running SAP application instances. The analysis layer holds optimization algorithms and techniques to train and evaluate performance models. Finally, a web server provides a user interface which supports different tasks of the PPSS process on the presentation layer. In its current implementation, the tasks to understand the environment, to characterize workloads, and to solve the service placement problem could be made available as part of a pilot phase in which eight employees from two data centers took the role of capacity management consumers. In order to evaluate the ease of use, the fidelity with real world phenomena, and the impact on the artifact environment, user feedback was collected in an online survey and discussed in two feedback workshops. Results indicate significant savings in time and effort when comparing the PPSS capacity management approach to the approach which is currently applied in the participating data centers. The possibility of defining placement constraints was rated to be essential. A lack of confidence with respect to the quality of service of computed solutions confirms the need to predict their performance. Finally, pilot users of both data centers emphasized their will to use the offered functions half-yearly to yearly.

5.2 Scientific contributions

In accordance with the research questions, raised in Section 1.2, the thesis makes the following contributions to the scientific knowledge base:

- Descriptive analysis of the utilization rate of standard transactions: Although COTS EAs offer standard business functions to a group of customers, products allow to develop additional business transactions to serve customer-specific purposes. In the domain of SAP EAs, the mean usage degree of these custom transactions lies around 20%, when considering the frequency of executions. Therefore, custom transactions are accountable for around 31.51% of the total CPU time. In general, large environments tend to show higher standardization degrees than small and medium environments. Results of this analysis are valuable for the design of any future research artifacts which follow the idea of collaboration in the domain of capacity management. From a practitioners perspective, information about standardization may serve as a baseline to estimate efforts of EA transformation projects.

- Descriptive analysis of server utilization levels in the domain of COTS EA: According to several studies, enterprise servers run at average utilization levels between 10% and 20%. However, in the domain of COTS EAs from SAP, mean utilization levels turn out to be lower. In fact, mean server utilization was measured to be around 8% over the last decade. This is the effect of a provisioning practice that is based on peak demands and the procurement of new servers for additional applications.

Results emphasize the relevance of future research artifacts which address the lack of data center efficiency along with its global climate consequences.

- Review and definition of placement constraint types: When placing services on servers, a number of individual constraints typically limit the solution space. This work analysis existing types of placement constraints using both the scientific literature and expert input from practitioners. As a result, ten placement constraint types were defined which allow to model individual realities as part of a service placement problem instantiation.

- Definition of criteria to classify service placement problems: Related approaches were classified in order to identify a research gap. Service placement problems, including VM placement problems, may be classified regarding the service state during solution deployment, the nature of workload profiles, the number of resource dimensions, the bin composition, and the consideration of placement constraints and service performance. Using these criteria, related consolidation approaches can be analyzed with special regard to their applicability in the domain of EAs.

- Review of existing performance prediction techniques: The work summarizes selection criteria for performance prediction techniques, which can be measurement-based, model-based or machine learning-based. Requirements, advantages, and drawbacks for each method help to select appropriate means for different prediction use cases in the domain of EA capacity management.

- Evaluation of machine learning-based prediction techniques which utilize shared performance counters for the domain of EAs: Black-box techniques require low amounts of domain expertise but are limited to observations included in the training data. The concept of sharing performance counters increases the volume and the variety of training data and makes alike techniques applicable already in the design phase. This work evaluates the applicability of machine learning techniques and proves bagging and boosting strategies to be sufficient ensemble learning methods in the tested scenario. In particular, Random forests and Boosted trees provide results with acceptable accuracy in the domain of capacity management. Accordingly, the performance of frequently used standard transactions could be predicted with a mean absolute percent error below 30%, using AdaBoost.

- Formulation of a multi-dimensional SPP: This work contributes a problem formulation for offline server consolidation efforts which supports multiple resource types, dynamic workload profiles, heterogeneous servers, and placement constraints.

- Evaluation of algorithms to solve the SPP: Four heuristics, two metaheuristics, and two hybrid algorithms were used to solve 12,384 realistic problem instances. If constraints exist, genetic algorithms perform significantly better than any other of the tested algorithms. A grouping genetic algorithm turned out to be sufficient for

unknown problem classes of the SPP as it depicts the problem of grouping services on available servers in the most natural way. Heuristics are highly efficient but lack to consider placement constraints. Experiment results contribute also to the related research fields of online server consolidation and load balancing.

- Design of a process model for EA capacity management: The research artifact of this work represents a method whose tasks are arranged in a process model which may be applied by practitioners and further adjusted by the scientific community. The method, termed PPSS, is based on a central knowledge base which holds monitoring data from the domain of a COTS EA and provides typical capacity management functions as a service. Therefore, it integrates a structured data mining process in order to build performance models. An integral part of PPSS is to balance operations costs and SLA compliance. It could be demonstrated how performance prediction results affect design recommendations with the objective to minimize the total costs of a data center. Therefore, the research artifact contributes to the claimed integration of activities from the field of APM and SPE.

With respect to the research goal, the designed artifact was evaluated to reduce costs of server consolidation efforts while increasing solution applicability. Both criteria are enabled by the centralization of core capacity management components and their maintenance, including monitoring data, algorithms, and performance models. Cost reductions result from automation and economies of scale. The centralization of performance models additionally enables to increase both volume and variety of training data which allows to apply black-box machine learning methods. This way, solution applicability benefits from performance evaluation while costly expert knowledge is avoided.

5.3 Limitations and future work

Enterprise applications utilize COTS software in order to save costs. The research artifact, described in this thesis, depends on a sufficient amount of training data which is formed by the APM output of these COTS EA. Therefore, the method is generally limited to widespread COTS EAs with high usage degrees of standard transactions. This requirement tends to be fulfilled predominantly in large data centers. Small and highly customized environments may not benefit enough from the designed functions. Furthermore, the method was evaluated using the example of SAP software. The application of machine learning-based techniques which utilize performance counters of alternative COTS EAs is subject to future research. With respect to the process model, a variant may evolve for native cloud applications. Here, the role of the capacity management provider may be taken by the software vendor itself, in order to minimize hosting costs.

The application area of the performance prediction, in its current design, is limited to the SPP solution candidates. Theses solution candidates were computed beforehand on the basis of historical workload profiles. As a benefit of this modular design, both the optimization and the prediction engine can be used separately. On the other hand, the

degrees of freedom for defining alternative load factors are limited. A tighter integration of performance predictions as part of the SPP could help to explore the solution space more effectively. Future research efforts could formulate a multi-objective optimization problem with the objective to minimize costs and maximize performance at the same time. Machine learning-based techniques, designed in this work, were proven to estimate transactional performance with acceptable accuracy in a matter of milliseconds. Hence, these may be integrated into the fitness function of a metaheuristic in order to make predictions for each population during the search phase. Related changes in the design would require to adjust the PPSS process model as load factors must be defined during the SPP instantiation.

Evaluation of the artifact is limited to a mathematical solution quality computation (SPP) and cross-validation using real test data (ML models). The actual deployment of a solution in a real data center could not be tested due to the organizational complexity of EA consolidation projects which typically affect the complete system landscape. Although the described approach represents the dominant evaluation strategy in the studied literature, it would be desirable to deploy a solution and measure its transactional performance in order to further evaluate practical applicability. In this regard, an interface of the PPSS to the subsequent service operations phase is to be explored in the future. The current method envisages results to be documented in the capacity plan so that recommended actions are technology-independent. However, real autonomous operations require to integrate solution deployment and monitor subsequent operations, resulting in a MAPE-K feedback loop for self-adaptive service operations.

Finally, evaluation on the level of EVAL 4 is limited to components of the PPSS for which a GUI was constructed. Therefore, the analyzed user feedback cannot be generalized to the tasks of performance prediction. Further development effort is required to complement the presentation layer for the remaining PPSS tasks and offer an extended interface to a group of test users. These efforts are planned for the near future of the ongoing research project.

With respect to the prediction accuracy, several improvements are conceivable. In this work, machine learning techniques were trained on the basis of hourly aggregates. Decreases of error metrics are expected if raw data for each dialog step was used. Due to data unavailability, this could not be tested in the course of this work but may be accomplished in a future research project. Furthermore, additional features which describe the workload and the resource capacity could be integrated in order to cover also the network and the storage layer of an EA. Finally, training data may be clustered, e.g., on the basis of their age or the software release. Then, specific models could be trained for each cluster with the objective to increase overall accuracy.

Hyperparameters of both machine learning models and metaheuristics were chosen on the basis of related work and sensitivity analyses. Hyperparameter optimization using a meta learning approach could help to further explore well-performing parameter sets for specific problem classes. For example, it was concluded that the GGA represents the best default choice for service placement problems with any set of placement constraints. A

meta learning system, however, could decide on the basis of case-specific input data, which algorithm is expected to provide highest solution quality in shortest time. Alike strategies become particularly relevant in online consolidation and load balancing problems where computation time must be kept low. The task to forecast workload evolution implicates further automation potential. In the current design of the PPSS, forecasts are made manually on the basis of the business strategy. If, however, major strategy-dependent workload changes are not expected in the future, forecasts could be made on the basis of expected workload patterns which result from trend analyses.

Further research is required on the matter of how to deal with infeasible constraint sets when solving the SPP. While few self-evident inconsistencies may be revealed by means of simple feasibility checks, full constraint compliance cannot always be guaranteed in advance. Therefore, individual weightings of the placement constraints could represent user priorities and help to direct the search process into favored areas. Such weights are supported by the design as they can be translated into varying penalty factors. In the evaluated implementation, constraint violations are penalized on the basis of the degree of violation. However, weights may further modify penalty factors and, thus, resolve states of conflicting constraints. Additional research is desired to reflect constraint priorities in penalization.

Finally, the main idea of the PPSS relies on the centralization of monitoring data in order to learn from observations made by similar customers. The PPSS could be extended by additional use cases which also benefit from this principle. For example, capacity planning (as opposed to capacity management) involves the challenge of sizing new resource components for expected workloads. Here, artifact components of this work would allow to optimize the SLA fulfillment, e.g., by identifying cost-efficient servers. Furthermore, benchmarking of EA performance requires to compare defined metrics against a norm. According to the ITIL publication on Continual service improvement, this norm may be defined on the basis of industry data from external organizations (Lloyd, 2011, p. 81). Therefore, the APM-KB could be extended to provide benchmarking functions and, if appropriate, to recommend actions to improve performance. For the latter, designed models may be utilized with varying load and capacity features in the future.

In its current state, the research artifact addresses the objective to evaluate cost saving potential of server consolidation efforts while increasing solution applicability. Therefore, the artifact was designed to fit the characteristics of EA environments along with their existing constraints, heterogeneity, seasonality, and SLAs. According to the research goal, the exceptional requirements on performance were addressed by cost-effective techniques. The research findings, presented throughout the thesis, are expected to push forward the use of computational intelligence in the domain of EA capacity management. Accompanying energy savings are required to contribute to the global climate protection. In the long term, balancing of costs and performance are to evolve from academic science to practical commodity in order to comply with the increasing complexity of the digital transformation.

A Performance model evaluation

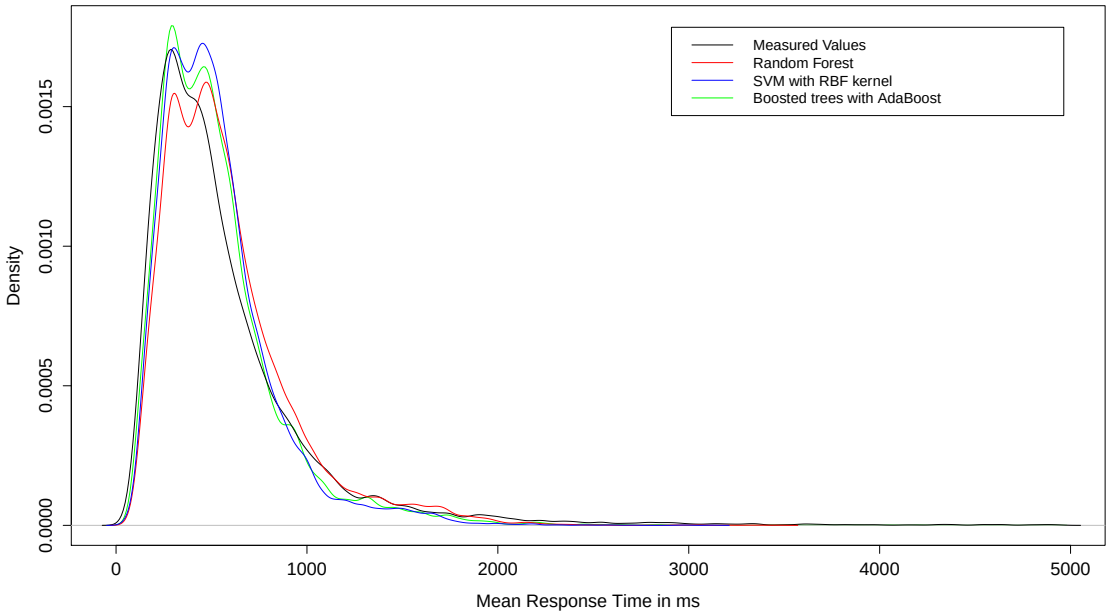

Figure A.1: Density plot for the transaction to change sales orders.

Business transaction	**Random forest**	**Boosted trees**
Create sales order	32.30	25.26
Change sales order	26.08	19.57
Display sales order	28.19	22.93
List of Sales Orders	26.44	21.95
Create Billing Document	39.04	28.66
Change Billing Document	38.97	29.34
Display Billing Document	33.74	25.75
Create Outbound Dlv. with Order Ref.	35.59	26.45
Change Outbound Delivery	36.25	27.01
Display Outbound Delivery	36.53	27.38

Table A.1: Mean absolute percent errors for the 10 most frequently used transactions.

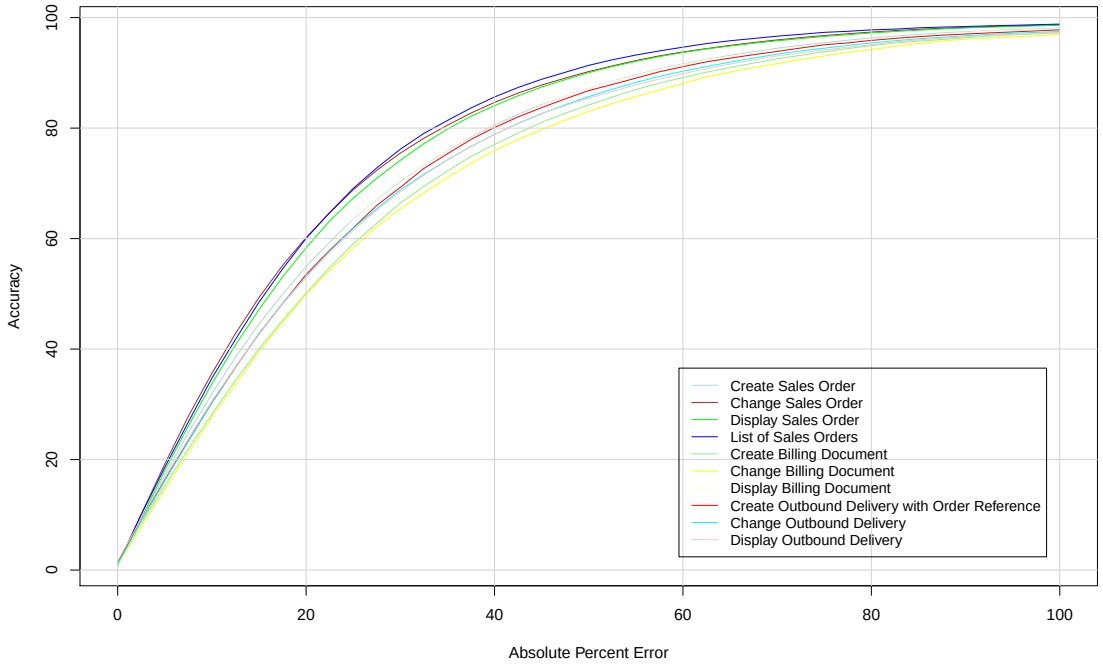

Figure A.2: REC curves for ten frequently used transaction types.

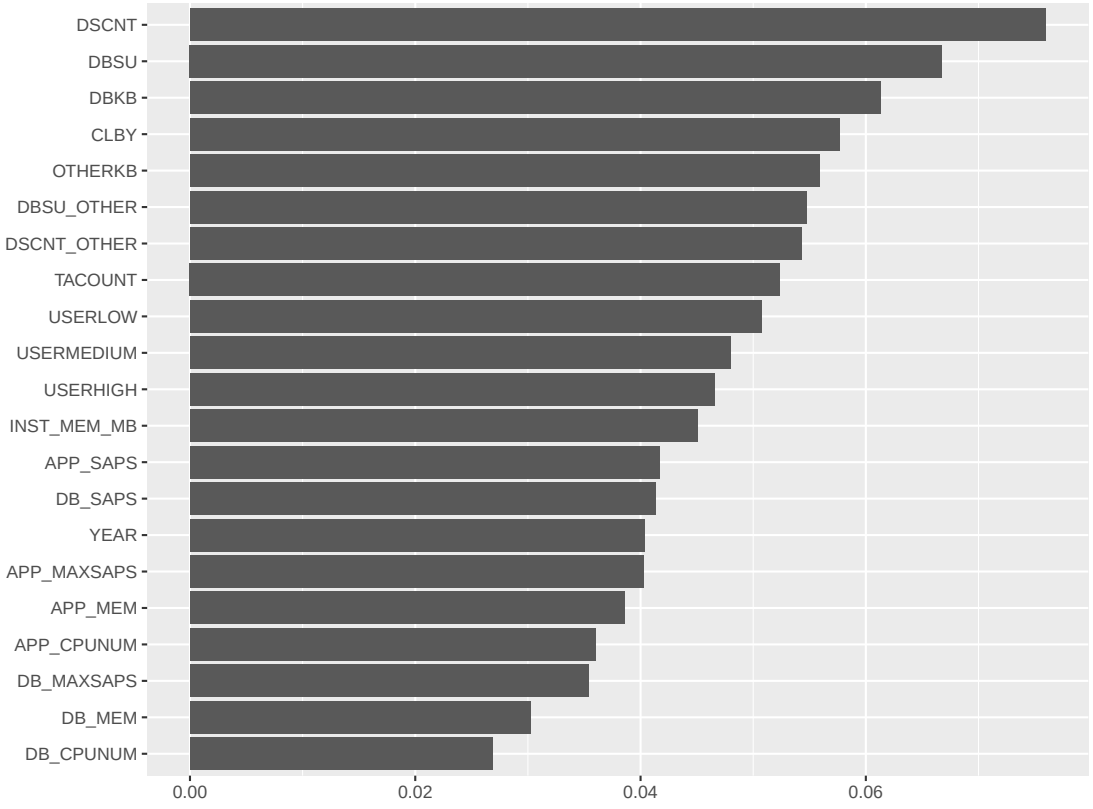

Figure A.3: Feature importance in percent as computed by AdaBoost.

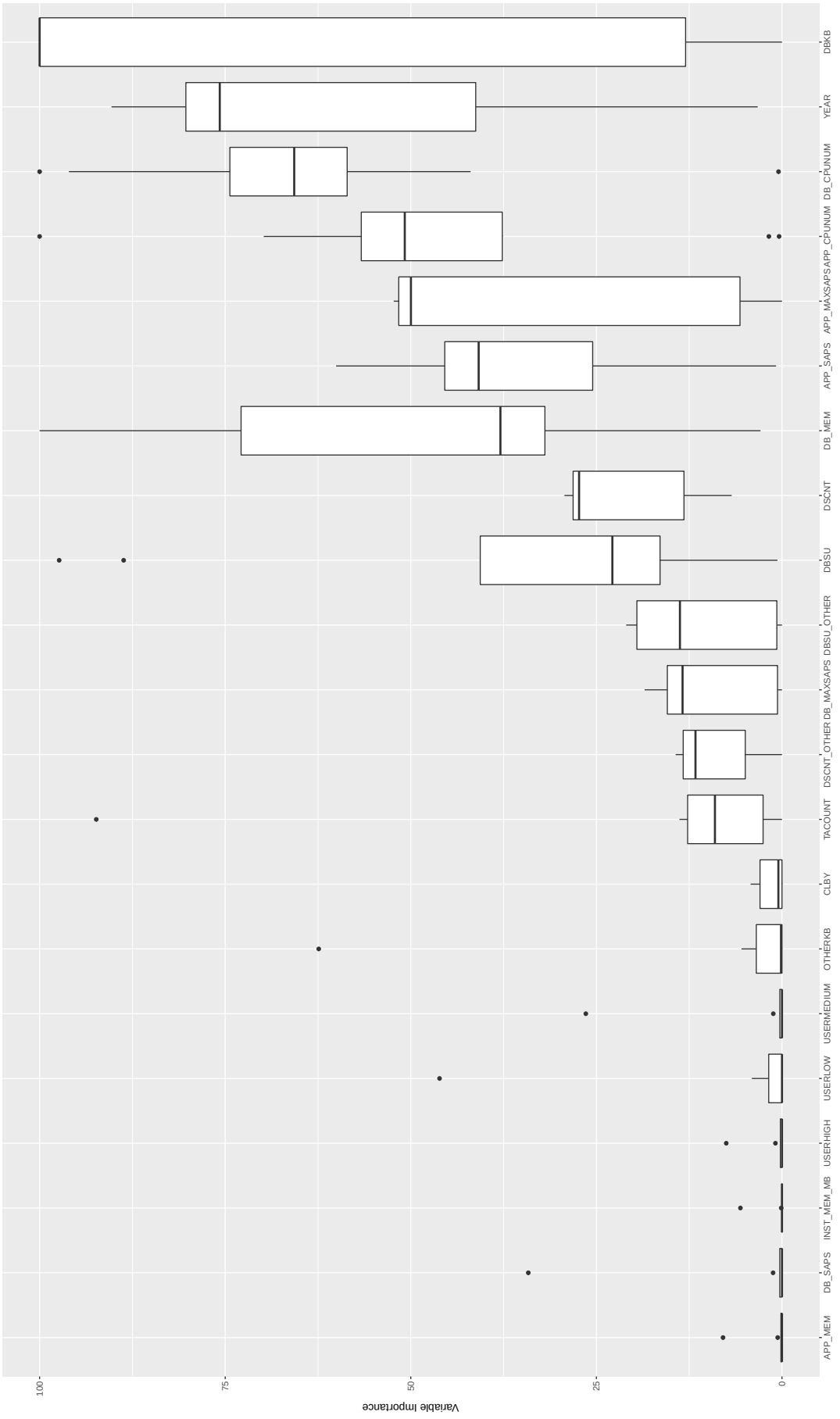

Figure A.4: Distribution of feature importance across feature selection techniques.

B Solution algorithm evaluation

Algorithm	Scenario	Mean constraint PF	Mean number of viol.	Mean PF C_{iso}	Mean PF C_{antico}	Mean PF $C_{extrares}$	Mean PF C_{util}
FFDsum	UC	0	0	0	0	0	0
FFDmax	UC	0	0	0	0	0	0
BFDsum	UC	0	0	0	0	0	0
BFDmax	UC	0	0	0	0	0	0
GA	UC	0	0	0	0	0	0
GGA	UC	0	0	0	0	0	0
GA_FF	UC	0	0	0	0	0	0
GA_BF	UC	0	0	0	0	0	0
FFDsum	C85	1.628333333	2.058139535	0.613565891	0.131860465	0.057848837	0.825271318
FFDmax	C85	1.611627907	2.067829457	0.616143411	0.149786822	0.061124031	0.784515504
BFDsum	C85	1.743391473	2.137596899	0.688081395	0.136414729	0.060387597	0.858817829
BFDmax	C85	1.782635659	2.182170543	0.687596899	0.177189922	0.06377907	0.854302326
GA	C85	0.219341085	0.539147287	0.087306202	0.002383721	0.008178295	0.121531008
GGA	C85	0.260251938	0.732170543	0.111065891	0.002635659	0.00996124	0.136763566
GA_FF	C85	0.942306202	1.614728682	0.509883721	0.060755814	0.024089147	0.347344961
GA_BF	C85	0.914573643	1.599031008	0.498100775	0.061724806	0.025562016	0.329341085
FFDsum	C65	7.05872093	2.178294574	0.613565891	0.131860465	0.057848837	6.255852713
FFDmax	C65	6.817906977	2.189922481	0.616143411	0.149786822	0.061124031	5.990813953
BFDsum	C65	7.200232558	2.253875969	0.688081395	0.136414729	0.060387597	6.315697674
BFDmax	C65	7.123624031	2.298449612	0.687596899	0.177189922	0.06377907	6.195193798
GA	C65	1.146976744	0.808333333	0.117403101	0.004806202	0.008430233	1.016356589
GGA	C65	1.40998062	1.013178295	0.159554264	0.008507752	0.010717054	1.231492248
GA_FF	C65	3.990949612	1.872674419	0.540232558	0.066589147	0.02872093	3.355387597
GA_BF	C65	3.843624031	1.863178295	0.54753876	0.06505814	0.030077519	3.201414729

Table B.1: Algorithm performance metrics w.r.t. constraint compliance.

Algorithm	Scenario	Mean fit_{diff}	Left 95% CI fit_{diff}	Right 95% CI fit_{diff}	Mean savings in percent	Mean run-time	Left 95% CI run-time	Right 95% CI run-time
FFDsum	UC	21.87810509	19.57518597	24.18102422	0.455346338	0.129844961	0.029343304	0.230346619
FFDmax	UC	22.10923033	19.68496483	24.53349583	0.456783946	0.257751938	0.116048964	0.399454912
BFDsum	UC	14.96400459	12.96542397	16.96258522	0.482186152	0.071705426	-0.000797712	0.144208565
BFDmax	UC	14.42270549	12.26082014	16.58459084	0.486181907	0.075581395	0.002958157	0.148204634
GA	UC	317.47848	277.6689056	357.2880545	0.67269554	860.2071705	780.0141165	940.4002246
GGA	UC	14.31527133	11.68563079	16.94491188	0.547217703	244.2635659	206.726069	281.8010628
GA_FF	UC	19.42017141	17.21966867	21.62067416	0.463865359	265.1507752	215.1726389	315.1289115
GA_BF	UC	11.07563613	9.219723719	12.93154855	0.495736797	276.1296512	229.2897625	322.9695399
FFDsum	C85	176.067236	158.2408829	193.8935891	0.455346338	0.215116279	0.176870626	0.253361932
FFDmax	C85	174.7280784	157.0594849	192.3966719	0.456783946	0.224806202	0.185662479	0.263949924
BFDsum	C85	185.9824576	169.3772465	202.5876687	0.482186152	0.215116279	0.17713879	0.253093768
BFDmax	C85	196.2715913	176.9905404	215.5526421	0.486181907	0.284883721	0.236917246	0.332850196
GA	C85	2.28566307	1.71242338	2.858902761	0.263939573	2219.820543	1983.802749	2455.838336
GGA	C85	5.613442047	4.597979695	6.628904399	0.277812899	500.505814	436.2608924	564.7507355
GA_FF	C85	84.44471031	74.72272635	94.16669426	0.438688953	369.1949612	306.5629202	431.8270023
GA_BF	C85	79.03490596	70.44667771	87.6231342	0.446183535	376.930814	317.7075765	436.1540514
FFDsum	C65	547.1944173	481.4303793	612.9584552	0.455346338	0.203488372	0.164121887	0.242854857
FFDmax	C65	529.7053158	466.6938773	592.7167542	0.456783946	0.281007752	0.209647876	0.352367628
BFDsum	C65	550.0169265	487.6155863	612.4182667	0.482186152	0.23255814	0.192361931	0.272754348
BFDmax	C65	552.4591014	487.8046518	617.1135509	0.486181907	0.269379845	0.223101374	0.315658316
GA	C65	0.629434267	0.315301529	0.943567005	0.211241525	2206.973062	1971.451075	2442.495049
GGA	C65	18.8889682	14.73298912	23.04494728	0.212998151	494.4517442	431.1867734	557.7167149
GA_FF	C65	266.0680868	228.6612812	303.4748923	0.428196962	363.5131783	302.385548	424.6408086
GA_BF	C65	242.05996	212.3971245	271.7227955	0.434451645	381.0802326	321.2543204	440.9061447

Table B.2: Algorithm performance metrics w.r.t. solution fitness and run-time.

C Graphical user interface

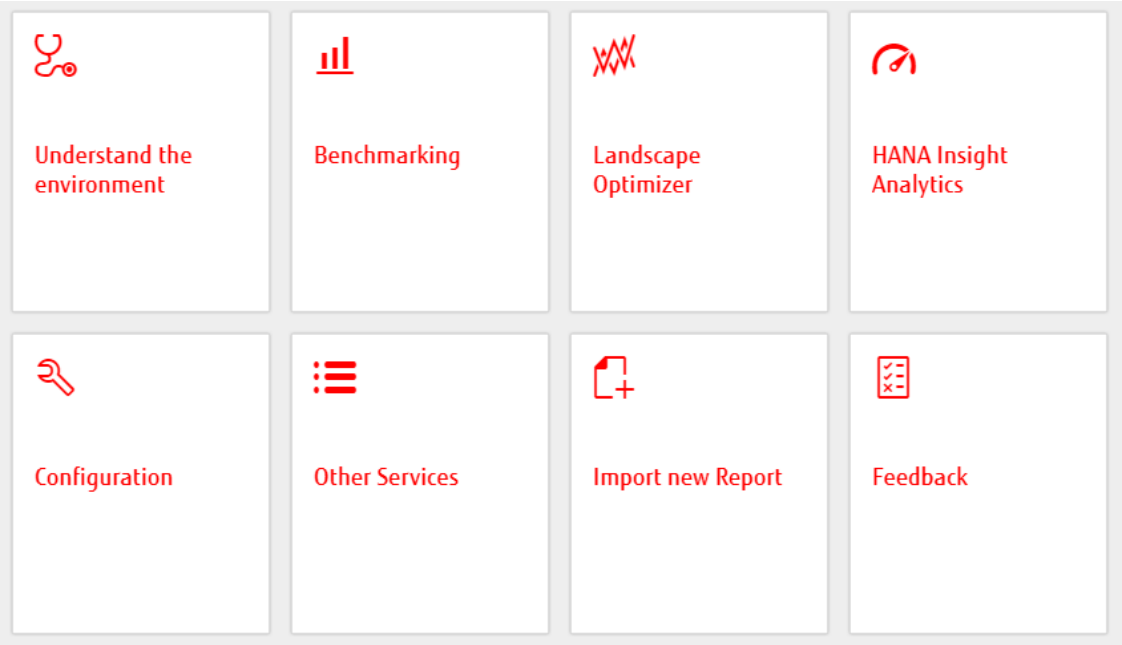

Figure C.1: Landing page of the APM-KB presentation layer.

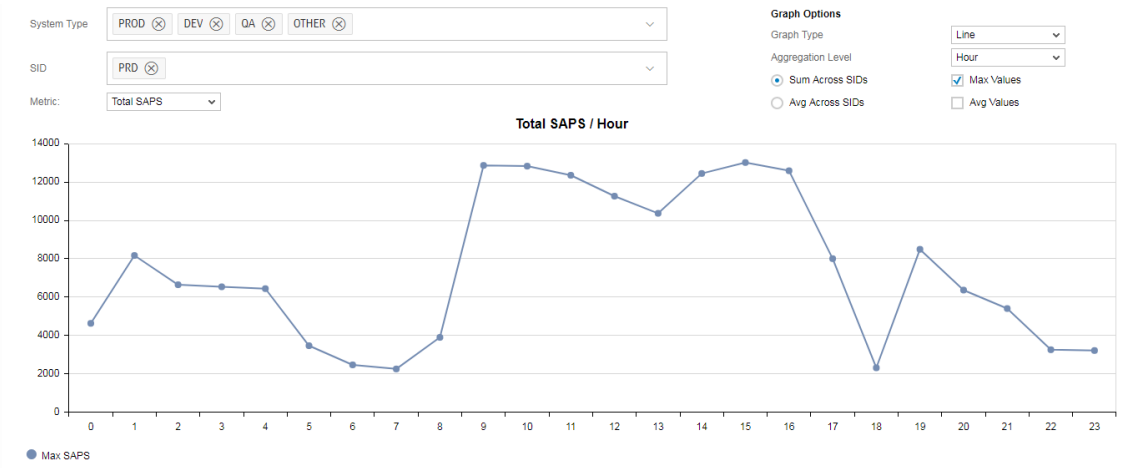

Figure C.2: Workload analysis from a service perspective.

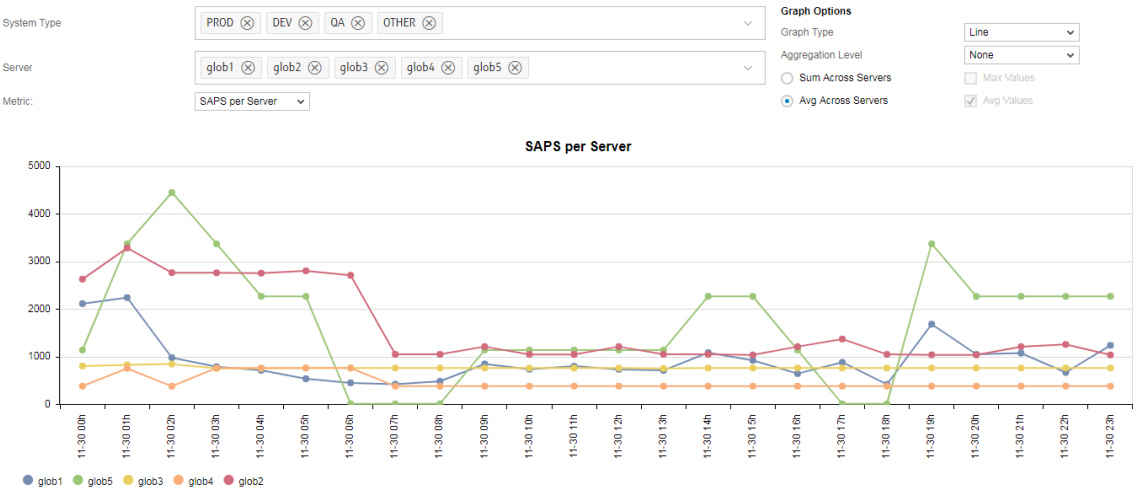

Figure C.3: Workload analysis from a server perspective.

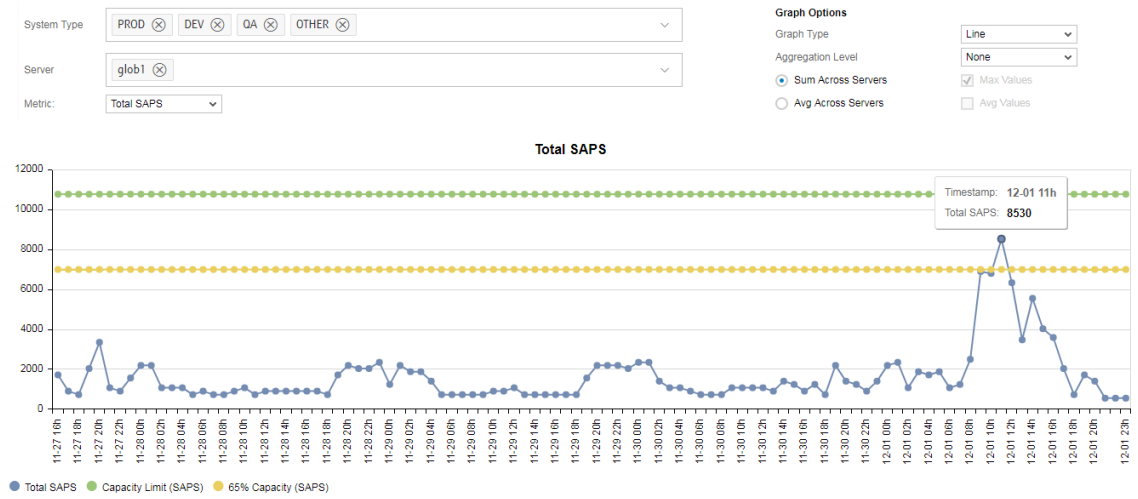

Figure C.4: Server utilization analysis.

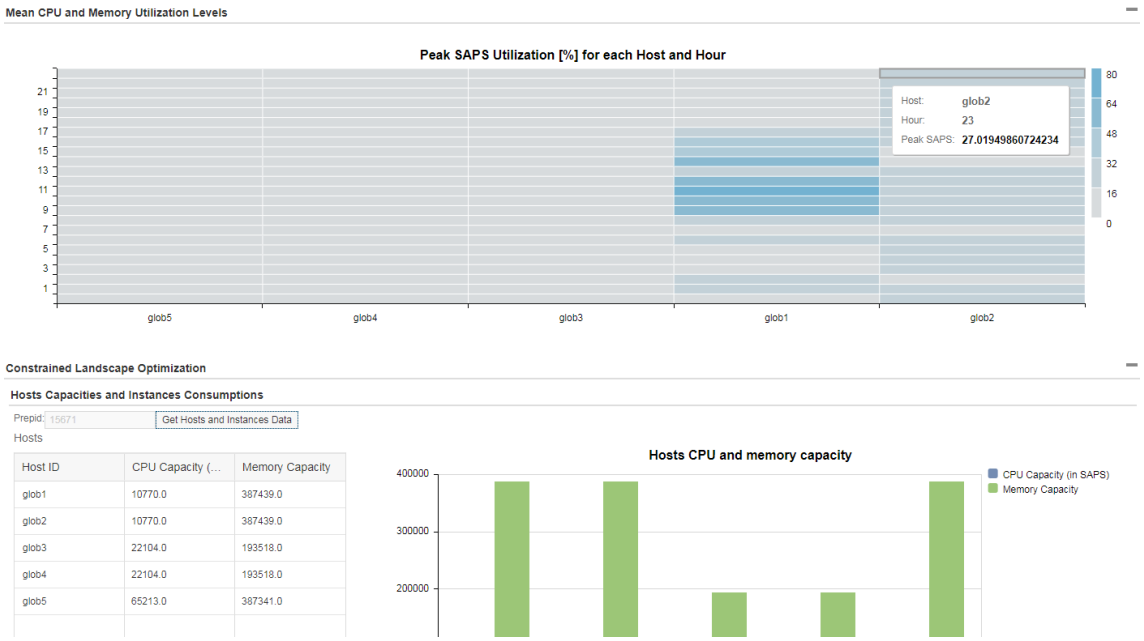

Figure C.5: Analysis of peak server utilization levels and capacity limits.

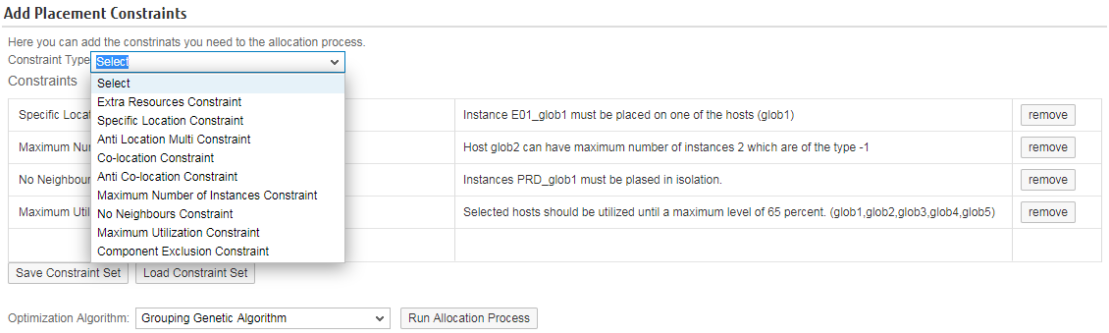

Figure C.6: Constraint modeling prior to optimization run.

Bibliography

Acton, M., Bertoldi, P., Booth, J., Newcombe, L., Rouyer, A., and Tozer, R. (2018). 2018 best practice guidelines for the eu code of conduct on data centre energy efficiency. JRC Technical Reports. (Cited on pages 29 and 30.)

Adamuthe, A. C., Pandharpatte, R. M., and Thampi, G. T. (2013). Multiobjective virtual machine placement in cloud environment. In Cloud & Ubiquitous Computing & Emerging Technologies (CUBE), 2013 International Conference on, pages 8–13. (Cited on pages 36, 66, and 107.)

Akhras, S. (2017). Applying placement constraints on server consolidation for enterprise application service providers. Master's thesis, Faculty of Computer Science, Otto von Guericke University Magdeburg. (Cited on pages 39, 87, and 139.)

Almeida, V. A. (2002). Capacity planning for web services techniques and methodology. In IFIP International Symposium on Computer Performance Modeling, Measurement and Evaluation, pages 142–157. (Cited on pages ix, 21, 22, 59, 79, 80, and 141.)

Almeida, V. A. and Menascé, D. A. (2002). Capacity planning an essential tool for managing web services. IT professional, 4(4):33–38. (Cited on page 79.)

Alvarez, V. (2015). C4 instance family certified for sap applications. URL https://aws.amazon.com/de/blogs/awsforsap/c4-instance-family-certified-for-sap-applications/. Last accessed: April 15, 2019. (Cited on page 133.)

Amazon (2019). Aws simple monthly calculator. URL https://calculator.s3.amazonaws.com/index.html. Last accessed: April 15, 2019. (Cited on page 132.)

Ankit, A., Lakshmi, J., and Nandy, S. (2013). Virtual machine placement optimization supporting performance slas. CloudCom 2013. (Cited on pages 30, 67, and 68.)

Appleby, K., Fakhouri, S., Fong, L., Goldszmidt, G., Kalantar, M., Krishnakumar, S., Pazel, D. P., Pershing, J., and Rochwerger, B. (2001). Oceano-sla based management of a computing utility. In Integrated Network Management Proceedings, 2001 IEEE/IFIP International Symposium on, pages 855–868. (Cited on pages 20 and 108.)

Bäck, T. (1996). Evolutionary algorithms in theory and practice. Oxford University Press. (Cited on pages 36 and 91.)

Bailey, M., Eastwood, M., Grieser, T., Borovick, L., Turner, V., and Gray, R. C. (2007).

Special study: Data center of the future. Technical report, International Data Corporation (IDC). (Cited on page 27.)

Baker, J. E. (1987). Reducing bias and inefficiency in the selection algorithm. In Proceedings of the second international conference on genetic algorithms, pages 14–21. (Cited on pages 38 and 92.)

Balsamo, S., Marco, A. D., Inverardi, P., and Simeoni, M. (2004). Model-based performance prediction in software development: A survey. IEEE Transactions on Software Engineering, 30(5):295–310. (Cited on page 47.)

Barnard, C. and Simon, H. A. (1947). Administrative behavior. a study of decision-making processes in administrative organization. New York, NY (USA) Free Press. (Cited on page 106.)

Barroso, L. A. and Hölzle, U. (2007). The case for energy-proportional computing. Computer, 12:33–37. (Cited on pages 3, 28, and 33.)

Baset, S. A. (2012). Cloud slas: present and future. ACM SIGOPS Operating Systems Review, 46(2):57–66. (Cited on page 20.)

Becker, S., Grunske, L., Mirandola, R., and Overhage, S. (2006). Performance prediction of component-based systems. In Architecting Systems with Trustworthy Components, pages 169–192. Springer. (Cited on pages 44 and 45.)

Becker, S., Koziolek, H., and Reussner, R. (2009). The palladio component model for model-driven performance prediction. Journal of Systems and Software, 82(1):3–22. (Cited on pages 45 and 49.)

Beloglazov, A., Abawajy, J., and Buyya, R. (2012). Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future generation computer systems, 28(5):755–768. (Cited on pages 2, 43, 60, 66, and 82.)

Beloglazov, A. and Buyya, R. (2010a). Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers. In Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science. (Cited on pages 3 and 28.)

Beloglazov, A. and Buyya, R. (2010b). Energy efficient resource management in virtualized cloud data centers. In 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing. (Cited on page 29.)

Bertoldi, P. (2018). Brochure: The european code of conduct for energy efficiency in data centres. URL https://ec.europa.eu/jrc/sites/jrcsh/files/eu_folder_code_of_conduct.pdf. Last accessed: January 23, 2019. (Cited on page 27.)

Bertolino, A. and Mirandola, R. (2004). Cb-spe tool: Putting component-based performance engineering into practice. In International Symposium on Component-Based Software Engineering, pages 233–248. (Cited on page 49.)

Bi, J. and Bennett, K. P. (2003). Regression error characteristic curves. In Proceedings of the 20th international conference on machine learning (ICML-03), pages 43–50. (Cited on page 103.)

Bichler, M., Setzer, T., and Speitkamp, B. (2006). Capacity planning for virtualized servers. In Workshop on Information Technologies and Systems (WITS), Milwaukee, WI, USA. (Cited on pages 34, 35, 45, 62, 65, and 70.)

Bin, E., Biran, O., Boni, O., Hadad, E., Kolodner, E. K., Moatti, Y., and Lorenz, D. H. (2011). Guaranteeing high availability goals for virtual machine placement. In Distributed Computing Systems (ICDCS), 2011 31st International Conference on, pages 700–709. (Cited on pages 40 and 41.)

Bishop, C. M. (2006). Pattern recognition and machine learning. springer. (Cited on page 99.)

Bobroff, N., Kochut, A., and Beaty, K. (2007). Dynamic placement of virtual machines for managing sla violations. In Integrated Network Management, 2007. IM'07. 10th IFIP/IEEE International Symposium on, pages 119–128. (Cited on pages 29, 30, 63, and 108.)

Boloor, K., Chirkova, R., Salo, T., and Viniotis, Y. (2010a). Heuristic-based request scheduling subject to a percentile response time sla in a distributed cloud. In Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE, pages 1–6. (Cited on pages 20, 65, 87, and 109.)

Boloor, K., Chirkova, R., Viniotis, Y., and Salo, T. (2010b). Dynamic request allocation and scheduling for context aware applications subject to a percentile response time sla in a distributed cloud. In Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on, pages 464–472. (Cited on pages 20 and 65.)

Bondarev, E., de With, P., Chaudron, M., and Muskens, J. (2005). Modelling of input-parameter dependency for performance predictions of component-based embedded systems. In 31st EUROMICRO Conference on Software Engineering and Advanced Applications, pages 36–43. (Cited on page 49.)

Bontempi, G. and Kruijtzer, W. (2002). A data analysis method for software performance prediction. In Proceedings of the conference on Design, automation and test in Europe, page 971. (Cited on page 79.)

Bosse, S. (2016). Optimierung der kosten und verfügbarkeit von it-dienstleistungen durch

lösung eines redundanz-allokation-problems. PhD thesis, Otto-von-Guericke-Universität Magdeburg, Fakultät für Informatik. (Cited on pages 34 and 36.)

Bosse, S., Hintsch, J., Schulz, C., Splieth, M., Müller, H., and Turowski, K. (2015). Evaluating it service design alternatives with respect to availability, response times and costs. In Proceedings of the 2nd HPI Cloud Symposium Operating the Cloud, pages 1–14. (Cited on page 11.)

Bosse, S., Müller, H., Akhras, S., and Turowski, K. (2019). Server consolidation for enterprise applications considering operational constraints. (Cited on pages 13 and 90.)

Braasch, B., Faustmann, A., Geringer, A., Müller, H., Neumann, K., Siegling, A., and Wegener, B. (2016). Sap hana administration. Rheinwerk Verlag. (Cited on page 13.)

Bray, D. and von Storch, H. (2009). Prediction or projection? the nomenclature of climate science. Science Communication, 30(4):534–543. (Cited on page 44.)

Breiman, L. (2001). Random forests. Machine learning, 45(1):5–32. (Cited on page 54.)

Breiman, L. and Cutler, A. (2019). Random forests. URL https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm. Last accessed: February 18, 2019. (Cited on pages 58 and 105.)

Brezulianu, A., Fira, M., and Fira, L. (2009). A genetic algorithm approach for a constrained employee scheduling problem as applied to employees at mall type shops. In Proceedings of the 2009 International Conference on Hybrid Information Technology, pages 497–501. (Cited on page 36.)

Brill, K. G. (2007). The invisible crisis in the data center: The economic meltdown of moores law. white paper, Uptime Institute, pages 2–5. (Cited on page 2.)

Brosig, F., Huber, N., and Kounev, S. (2014). Architecture-level software performance abstractions for online performance prediction. Science of Computer Programming, 90:71–92. (Cited on pages 45 and 47.)

Brown, R., Masanet, E., Nordman, B., Tschudi, B., Shehabi, A., Stanley, J., Koomey, J., Sartor, D., and Chan, P. (2007). Report to congress on server and data center energy efficiency: Public law 109-431. Technical report, U.S. Environmental Protection Agency. (Cited on pages 27 and 65.)

Browne, S., Dongarra, J., Garner, N., London, K., and Mucci, P. (2000). A scalable cross-platform infrastructure for application performance tuning using hardware counters. In SC'00: Proceedings of the 2000 ACM/IEEE Conference on Supercomputing, pages 42–42. (Cited on page 78.)

Brownlee, J. (2016). Boosting and adaboost for machine learning. URL https://machinelearningmastery.com/boosting-and-adaboost-for-machine-learning/. Last accessed: April 25, 2019. (Cited on page 56.)

Brunnert, A. and Krcmar, H. (2015). Continuous performance evaluation and capacity planning using resource profiles for enterprise applications. Journal of Systems and Software, 123:239–262. (Cited on pages 4 and 46.)

Brunnert, A., van Hoorn, A., Willnecker, F., Danciu, A., Hasselbring, W., Heger, C., Herbst, N., Jamshidi, P., Jung, R., von Kistowski, J., et al. (2015). Performance-oriented devops: A research agenda. arXiv:1508.04752. (Cited on pages 4, 16, 17, 20, 21, 26, and 44.)

Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. Data mining and knowledge discovery, 2(2):121–167. (Cited on page 55.)

Calcavecchia, N. M., Biran, O., Hadad, E., and Moatti, Y. (2012). Vm placement strategies for cloud scenarios. In Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on, pages 852–859. (Cited on pages 39, 41, and 68.)

Cannon, D., Wheeldon, D., Lacy, S., and Hanna, A. (2011). Itil service strategy. Tso London. (Cited on page 20.)

Cao, Z. and Dong, S. (2014). An energy-aware heuristic framework for virtual machine consolidation in cloud computing. The Journal of Supercomputing, 69(1):429–451. (Cited on page 107.)

Carr, N. G. (2005). The end of corporate computing. MIT Sloan Management Review, 46(3):67. (Cited on pages 27, 65, and 105.)

Caruana, R. and Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. In Proceedings of the 23rd international conference on Machine learning, pages 161–168. (Cited on pages 57, 99, and 100.)

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., and Wirth, R. (2000). Crisp-dm 1.0 step-by-step data mining guide. resreport, The CRISP-DM consortium. (Cited on pages ix and 57.)

Chaudhuri, S., Narasayya, V., and Ramamurthy, R. (2004). Estimating progress of execution for sql queries. In Proceedings of the 2004 ACM SIGMOD international conference on Management of data, pages 803–814. (Cited on page 49.)

Chen, S., Gorton, I., Liu, A., and Liu, Y. (2002). Performance prediction of cots component-based enterprise applications. In ICSE Workshop, 5th CBSE. (Cited on pages 44, 45, 46, and 84.)

Chen, S., Liu, Y., Gorton, I., and Liu, A. (2005). Performance prediction of component-based applications. Journal of Systems and Software, 74(1):35–43. (Cited on pages 44, 45, 46, and 49.)

Cherkasova, L., Ozonat, K., Mi, N., Symons, J., and Smirni, E. (2009). Automated anomaly detection and performance modeling of enterprise applications. ACM Transactions on Computer Systems (TOCS), 27(3):6. (Cited on pages 15, 19, 46, 49, 62, 94, and 95.)

Cherkasova, L. and Rolia, J. (2006). R-opus: A composite framework for application performability and qos in shared resource pools. In International Conference on Dependable Systems and Networks. (Cited on pages 2, 5, 62, and 82.)

Coit, D. and Smith, A. (1996). Reliability optimization of series-parallel systems using a genetic algorithm. IEEE Transactions on Reliability, 45:254–266. (Cited on pages 87, 91, and 92.)

Collins, A. (2018). The global risks report 2018 - 13th edition. Technical report, World Economic Forum. (Cited on page 2.)

Cortes, C. and Vapnik, V. (1995). Support-vector networks. Machine learning, 20(3):273–297. (Cited on pages 54 and 55.)

Criminisi, A., Shotton, J., Konukoglu, E., et al. (2012). Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. Foundations and Trends® in Computer Graphics and Vision, 7(2–3):81–227. (Cited on page 54.)

Dang, H. T. and Hermenier, F. (2013). Higher sla satisfaction in datacenters with continuous vm placement constraints. In Proceedings of the 9th Workshop on Hot Topics in Dependable Systems, page 1. (Cited on pages 3, 39, 40, and 41.)

Daniels, J. (2009). Server virtualization architecture and implementation. Crossroads, 16(1):8–12. (Cited on page 30.)

Davis, J. and LaMonica, M. (1998). Scooping up vanilla erp. InfoWorld, 20(47):1–3. (Cited on pages 17 and 19.)

Dittrich, Y., Vaucouleur, S., and Giff, S. (2009). Erp customization as software engineering: knowledge sharing and cooperation. IEEE software, 26(6). (Cited on page 6.)

Drucker, H. (1997). Improving regressors using boosting techniques. In ICML, pages 107–115. (Cited on pages 56, 101, and 105.)

Drucker, H., Burges, C. J., Kaufman, L., Smola, A. J., and Vapnik, V. (1997). Support

vector regression machines. In Advances in neural information processing systems, pages 155–161. (Cited on page 54.)

Duan, R., Nadeem, F., Wang, J., Zhang, Y., Prodan, R., and Fahringer, T. (2009). A hybrid intelligent method for performance modeling and prediction of workflow activities in grids. In Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, pages 339–347. (Cited on pages 47, 49, and 99.)

Ebert, C. (2014). Systematisches requirements engineering: Anforderungen ermitteln, dokumentieren, analysieren und verwalten. dpunkt. verlag. (Cited on pages ix and 76.)

Falkenauer, E. and Delchambre, A. (1992). A genetic algorithm for bin packing and line balancing. In Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on, pages 1186–1192. (Cited on page 91.)

Fan, X., Weber, W.-D., and Barroso, L. A. (2007). Power provisioning for a warehouse-sized computer. In ACM SIGARCH computer architecture news, pages 13–23. (Cited on pages 28 and 33.)

Feller, E., Rilling, L., and Morin, C. (2011). Energy-aware ant colony based workload placement in clouds. In Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing. (Cited on pages 65, 66, 68, and 71.)

Filani, D., He, J., Gao, S., Rajappa, M., Kumar, A., Shah, P., and Nagappan, R. (2008). Dynamic data center power management: Trends, issues, and solutions. Intel Technology Journal, 12(1). (Cited on pages 2 and 32.)

Freitas, N. d. (2013). Machine learning - random forests, university of british columbia. URL https://www.youtube.com/watch?v=DHspIG64CVM. Last accessed: February 16, 2019. (Cited on page 54.)

Freitas, N. d. (2015). Linear regression, university of oxford. URL https://www.youtube.com/watch?v=DHspIG64CVM. Last accessed: February 15, 2019. (Cited on pages ix, 50, and 51.)

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. Journal of computer and system sciences, 55(1):119–139. (Cited on page 56.)

Frey, C. B. and Osborne, M. A. (2017). The future of employment: how susceptible are jobs to computerisation? Technological forecasting and social change, 114:254–280. (Cited on page 1.)

Friedman, J., Hastie, T., and Tibshirani, R. (2017). The elements of statistical learning. Springer series in statistics New York. (Cited on pages 57, 100, and 105.)

Fujitsu (2012). Software optimization services sap - system inspection service. URL https://www.fujitsu.com/us/Images/Services-SAP-System-Inspection-Software-Optimization-factsheet.pdf. Last accessed: January 17, 2019. (Cited on page 22.)

Fujitsu (2018). Fujitsu integrated system primeflex for sap landscapes. URL https://www.fujitsu.com/de/solutions/infrastructure/dynamic-infrastructure/flexframe/. Last accessed: January 25, 2019. (Cited on page 31.)

Ganapathi, A., Kuno, H., Dayal, U., Wiener, J. L., Fox, A., Jordan, M., and Patterson, D. (2009). Predicting multiple metrics for queries: Better decisions enabled by machine learning. In Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on, pages 592–603. (Cited on pages 43, 48, and 49.)

Gao, Y., Guan, H., Qi, Z., Hou, Y., and Liu, L. (2013). A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. Journal of Computer and System Sciences, 79(8):1230–1242. (Cited on pages 29, 32, and 66.)

Gartner (2009). 10 key actions to reduce it infrastructure and operations cost structure. URL http://www.icomm.co.uk/getattachment/31d8d608-b18f-473a-8f7a-92b8e335c52a/Gartner-10-Key-Actions-To-Reduce-IT-Infrastucture.aspx. Last accessed: January 24, 2019. (Cited on pages 29 and 34.)

Gartner (2011). Ten key actions to reduce it infrastructure and operations costs. URL http://www.gartner.com/newsroom/id/1807615. (Cited on page 28.)

Gartner (2017). Gartner it glossary. URL https://www.gartner.com/it-glossary/application-performance-monitoring-apm. Last accessed: January 14, 2019. (Cited on page 4.)

Gartner (2018a). Gartner survey shows organizations are slow to advance in data and analytics. URL https://www.gartner.com/en/newsroom/press-releases/2018-02-05-gartner-survey-shows-organizations-are-slow-to-advance-in-data-and-analytics. Last accessed: January 14, 2019. (Cited on pages 4 and 78.)

Gartner (2018b). Worldwide it spending forecast. URL https://www.gartner.com/en/newsroom/press-releases/2018-10-17-gartner-says-global-it-spending-to-grow-3-2-percent-in-2019. Last accessed: January 21, 2019. (Cited on page 2.)

Gerdes, I., Klawonn, F., and Kruse, R. (2013). Evolutionäre algorithmen: Genetische algorithmen, strategien und optimierungsverfahren, beispielanwendungen. Springer-Verlag. (Cited on page 37.)

Gimpel, H. and Röglinger, M. (2015). Digital transformation: changes and chances–insights based on an empirical study. resreport, Fraunhofer Institute for Applied Information Technology. (Cited on page 1.)

Gmach, D., Krompass, S., Scholz, A., Wimmer, M., and Kemper, A. (2008). Adaptive quality of service management for enterprise services. ACM Transactions on the Web (TWEB), 2(1):8. (Cited on pages 20, 31, 44, 64, 70, 71, 93, 94, 107, 108, 120, and 133.)

Gneiting, T. (2011). Making and evaluating point forecasts. Journal of the American Statistical Association, 106(494):746–762. (Cited on page 58.)

Goldberg, R. P. (1974). Survey of virtual machine research. Computer, 7(6):34–45. (Cited on page 30.)

Goudarzi, H., Ghasemazar, M., and Pedram, M. (2012). Sla-based optimization of power and migration cost in cloud computing. In Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012), pages 172–179. (Cited on pages 1 and 94.)

Grinshpan, L. (2012). Solving enterprise applications performance puzzles: queuing models to the rescue. John Wiley & Sons. (Cited on pages 2, 3, 15, and 20.)

Gupta, C., Mehta, A., and Dayal, U. (2008). Pqr: Predicting query execution times for autonomous workload management. In Autonomic Computing, 2008. ICAC'08. International Conference on, pages 13–22. (Cited on pages 48, 49, and 99.)

Hallawi, H., Mehnen, J., and He, H. (2017). Multi-capacity combinatorial ordering ga in application to cloud resources allocation and efficient virtual machines consolidation. Future Generation Computer Systems, 69:1–10. (Cited on pages 67 and 92.)

Helbig, M. and Engelbrecht, A. P. (2013). Analysing the performance of dynamic multi-objective optimisation algorithms. In Evolutionary Computation (CEC), 2013 IEEE Congress on, pages 1531–1539. (Cited on page 34.)

Hermenier, F., Lawall, J., and Muller, G. (2013). Btrplace: A flexible consolidation manager for highly available applications. IEEE Transactions on dependable and Secure Computing, page 1. (Cited on pages 41, 42, 67, 71, 80, and 129.)

Hevner, A. R., March, S. T., Park, J., and Ram, S. (2004). Design Science in Information Systems Research. Management Information Systems Quarterly, 28(1):75–105. (Cited on pages ix, 5, 6, 7, 8, 9, 10, 87, 113, 114, and 120.)

Heyme, R. and Menge, A. M. (2017). Digitalisierung in sachsen-anhalt erfolgreich gestalten. URL https://library.fes.de/pdf-files/bueros/sachsen-anhalt/13749.pdf. Last accessed: January 14, 2019. (Cited on page 1.)

Hofmann, M. (2006). Support vector machines-kernels and the kernel trick. Notes, 26. (Cited on page 101.)

Holland, C. P. and Light, B. (1999). A critical success factors model for erp implementation. IEEE software, pages 30–36. (Cited on page 6.)

Hong, H.-J., Chen, D.-Y., Huang, C.-Y., Chen, K.-T., and Hsu, C.-H. (2013). Qoe-aware virtual machine placement for cloud games. In 2013 12th Annual Workshop on Network and Systems Support for Games (NetGames), pages 1–2. (Cited on page 66.)

Hong, K.-K. and Kim, Y.-G. (2002). The critical success factors for erp implementation: an organizational fit perspective. Information & management, 40(1):25–40. (Cited on page 6.)

Horkỳ, V., Libič, P., Marek, L., and Steinhauser, A. (2015). Utilizing performance unit tests to increase performance awareness and tuma, petr. In Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering, pages 289–300. (Cited on page 47.)

Huang, L., Jia, J., Yu, B., Chun, B.-G., Maniatis, P., and Naik, M. (2010). Predicting execution time of computer programs using sparse polynomial regression. In Advances in neural information processing systems, pages 883–891. (Cited on pages 47, 49, and 99.)

Hunnebeck, L., Rudd, C., Lacy, S., and Hanna, A. (2011). Itil service design. The Stationery Office (TSO). (Cited on pages 4, 20, 22, 23, 24, 25, 32, 43, 44, 57, 70, 71, 77, 78, 82, 83, 93, 106, and 109.)

Hyser, C., McKee, B., Gardner, R., and Watson, B. J. (2007). Autonomic virtual machine placement in the data center. Hewlett Packard Laboratories, Tech. Rep. HPL-2007-189, 189. (Cited on pages 3, 30, 32, 33, 39, 62, 68, 76, 82, and 108.)

IBM (2005). An architectural blueprint for autonomic computing. Technical report, IBM. (Cited on page 80.)

IBM (2016). Analytics solutions unified method - implementations with agile principles. URL ftp://ftp.software.ibm.com/software/data/sw-library/services/ASUM.pdf. Last accessed: February 18, 2019. (Cited on page 57.)

IT-Onlinemagazin (2015). Users rate sap-innovations: Survey result 2015. URL http://it-onlinemagazin.de/anwender-bewerten-sap-innovationen-umfrageergebnisse-2015/. Last accessed: January 25, 2019. (Cited on page 31.)

Jammal, M., Kanso, A., and Shami, A. (2015). High availability-aware optimization digest for applications deployment in cloud. In ICC, pages 6822–6828. (Cited on pages 40 and 41.)

Janssen, S. and Marquard, U. (2007). Sizing sap systems. SAP PRESS. (Cited on pages 28, 85, 108, and 119.)

Jennings, B. and Stadler, R. (2015). Resource management in clouds: Survey and research challenges. Journal of Network and Systems Management, 23:567–619. (Cited on pages 34 and 65.)

Jin, Y., Wen, Y., Chen, Q., and Zhu, Z. (2013). An empirical investigation of the impact of server virtualization on energy efficiency for green data center. The Computer Journal, 56(8):977–990. (Cited on page 43.)

Johnson, P. and Marker, T. (2009). Data centre energy efficiency product profile. Pitt & Sherry, report to equipment energy efficiency committee (E3) of The Australian Government Department of the Environment, Water, Heritage and the Arts (DEWHA). (Cited on pages 1, 2, 27, 28, 30, and 65.)

Jordan, M. I. (2004). The kernel trick. URL https://people.eecs.berkeley.edu/~jordan/courses/281B-spring04/lectures/lec3.pdf. Last accessed: February 18, 2019. (Cited on pages ix and 55.)

JRC (2008). Code of conduct on data centres energy efficiency - version 1.0. URL http://ec.europa.eu/information_society/activities/sustainable_growth/docs/datacenter_code-conduct.pdf. Last accessed: January 23, 2019. (Cited on page 27.)

Kaplan, J. M., Forrest, W., and Kindler, N. (2008). Revolutionizing data center energy efficiency. Technical report, Technical report, McKinsey & Company. (Cited on page 3.)

Kappler, T., Koziolek, H., Krogmann, K., and Reussner, R. H. (2008). Towards automatic construction of reusable prediction models for component-based performance engineering. Software Engineering, 121:140–154. (Cited on page 46.)

Keerthi, S. S. and Lin, C.-J. (2003). Asymptotic behaviors of support vector machines with gaussian kernel. Neural computation, 15(7):1667–1689. (Cited on page 101.)

Kégl, B. (2013). The return of adaboost. mh: multi-class hamming trees. arXiv preprint arXiv:1312.6086. (Cited on page 57.)

Koomey, J. (2011). Growth in data center electricity use 2005 to 2010. Technical report, Analytics Press. (Cited on pages 2 and 30.)

Koomey, J. and Taylor, J. (2015). New data supports finding that 30 percent of servers are comatose, indicating that nearly a third of capital in enterprise data centers is wasted. June, 3:2015. (Cited on page 3.)

Koomey, J. and Taylor, J. (2017). Zombie/comatose servers redux. Technical report,

Stanford University, Anthesis, Koomey Analytics. (Cited on page 28.)

Kowarschick, C. (2019). Sap benchmark. URL https://www.xware-gmbh.de/benchmark.html. Last accessed: February 6, 2019. (Cited on page 84.)

Krogmann, K., Kuperberg, M., and Reussner, R. (2010). Using genetic search for reverse engineering of parametric behavior models for performance prediction. IEEE Transactions on Software Engineering, 36(6):865–877. (Cited on pages 45 and 46.)

Kruse, R., Borgelt, C., Braune, C., Mostaghim, S., and Steinbrecher, M. (2016). Computational intelligence: a methodological introduction. Springer. (Cited on pages ix, 37, and 38.)

Kulturel-Konak, S., Smith, A. E., and Coit, D. W. (2003). Efficiently solving the redundancy allocation problem using tabu search. IIE transactions, 35(6):515–526. (Cited on page 86.)

Kuperberg, M., Krogmann, K., and Reussner, R. (2008). Performance prediction for black-box components using reengineered parametric behaviour models. In International Symposium on Component-Based Software Engineering, pages 48–63. (Cited on page 46.)

Kusic, D., Kephart, J. O., Hanson, J. E., Kandasamy, N., and Jiang, G. (2009). Power and performance management of virtualized computing environments via lookahead control. Cluster computing, 12(1):1–15. (Cited on pages 15, 28, 31, 63, and 71.)

Larrañaga, P., Kuijpers, C. M. H., Murga, R. H., Inza, I., and Dizdarevic, S. (1999). Genetic algorithms for the travelling salesman problem: A review of representations and operators. Artificial Intelligence Review, 13(2):129–170. (Cited on page 93.)

Lasi, H., Fettke, P., Kemper, H.-G., Feld, T., and Hoffmann, M. (2014). Industry 4.0. Business & Information Systems Engineering, 6(4):239–242. (Cited on page 1.)

Lee, J., Kao, H.-A., and Yang, S. (2014). Service innovation and smart analytics for industry 4.0 and big data environment. Procedia Cirp, 16:3–8. (Cited on page 1.)

Leff, A. and Rayfield, J. T. (2001). Web-application development using the model/view/-controller design pattern. In Proceedings fifth ieee international enterprise distributed object computing conference, pages 118–127. (Cited on page 139.)

Lewis, R. (2009). A general-purpose hill-climbing method for order independent minimum grouping problems: A case study in graph colouring and bin packing. Computers & Operations Research, 36:2295–2310. (Cited on pages 34 and 92.)

Li, Q. and Bauer, M. (2005). Understanding the performance of enterprise applications. In

Systems, Man and Cybernetics, 2005 IEEE International Conference on, pages 2825–2829. (Cited on pages 15 and 16.)

Liaw, A. and Wiener, M. (2018). Breiman and cutler's random forests for classification and regression - version 4.6-14. URL https://cran.r-project.org/web/packages/randomForest/randomForest.pdf. Last accessed: February 16, 2019. (Cited on pages 54 and 100.)

Lilja, D. J. (2005). Measuring computer performance: a practitioner's guide. Cambridge university press. (Cited on page 49.)

Liu, D. S., Tan, K. C., Huang, S. Y., Goh, C. K., and Ho, W. K. (2008). On solving multiobjective bin packing problems using evolutionary particle swarm optimization. European Journal of Operational Research, 190:357–382. (Cited on pages 34 and 92.)

Liu, Y., Fekete, A., and Gorton, I. (2004). Predicting the performance of middleware-based applications at the design level. In ACM SIGSOFT Software Engineering Notes, pages 166–170. (Cited on page 45.)

Lloyd, V. (2011). Itil continual service improvement. The Stationery Office (TSO). (Cited on page 156.)

Loibl, A. (2015). Sap sizing tool erklärt. URL https://dafrk-blog.com/de/sap-sizing-tool-erklaert/. Last accessed: March 28, 2019. (Cited on page 108.)

López-Pires, F. and Báran, B. (2015). Virtual machine placement literature review. arXiv preprint arXiv:1506.01509. (Cited on pages ix, 3, 32, 33, 34, 38, 39, 59, 60, 61, 71, 72, 85, and 89.)

March, S. T. and Smith, G. F. (1995). Design and natural science research on information technology. Decision support systems, 15(4):251–266. (Cited on page 6.)

Matsunaga, A. and Fortes, J. A. (2010). On the use of machine learning to predict the time and resources consumed by applications. In Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pages 495–504. (Cited on pages 43, 48, 83, 95, 99, and 103.)

Matt, C., Hess, T., and Benlian, A. (2015). Digital transformation strategies. Business & Information Systems Engineering, 57(5):339–343. (Cited on page 1.)

Mayer, D. and Butler, D. (1993). Statistical validation. Ecological modelling, 68(1-2):21–32. (Cited on page 58.)

Menascé, D. A. (2003). Automatic qos control. IEEE Internet Computing, 7(1):92–95. (Cited on page 44.)

Menascé, D. A. (2004). Composing web services: A qos view. Internet Computing, IEEE, 8(6):88–90. (Cited on pages 47 and 49.)

Menascé, D. A. and Almeida, V. A. (2002). Capacity planning for web services: metrics, models, and methods. Prentice Hall PTR Upper Saddle River, NJ. (Cited on page 49.)

Menascé, D. A., Almeida, V. A., Dowdy, L. W., and Dowdy, L. (2004). Performance by design: computer capacity planning by example. Prentice Hall Professional. (Cited on pages 21, 22, 79, and 97.)

Menascé, D. A. and Ngo, P. (2009). Understanding cloud computing: Experimentation and capacity planning. In Int. CMG Conference. (Cited on page 45.)

MGET (2019). Fit random forest model. URL http://code.env.duke.edu/projects/mget/export/HEAD/MGET/Trunk/PythonPackage/dist/TracOnlineDocumentation/Documentation/ArcGISReference/RandomForestModel.FitToArcGISTable.html. Last accessed: March 25, 2019. (Cited on page 100.)

Mi, H., Wang, H., Yin, G., Zhou, Y., Shi, D., and Yuan, L. (2010). Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers. In IEEE International Conference on Services Computing (SCC), Miami, FL, USA, pages 514–521. (Cited on page 28.)

Mi, N., Casale, G., Cherkasova, L., and Smirni, E. (2008). Burstiness in multi-tier applications: Symptoms, causes, and new models. In ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing, pages 265–286. (Cited on page 106.)

Miller, B. L., Goldberg, D. E., et al. (1995). Genetic algorithms, tournament selection, and the effects of noise. Complex systems, 9(3):193–212. (Cited on pages 37 and 92.)

Mills, K., Filliben, J., and Dabrowski, C. (2011). Comparing vm-placement algorithms for on-demand clouds. In Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on, pages 91–98. (Cited on page 38.)

Minas, L. and Ellison, B. (2009). Energy efficiency for information technology: How to reduce power consumption in servers and data centers. Intel Press. (Cited on page 2.)

Mohamadi Bahram Abadi, R., Rahmani, A. M., and Alizadeh, S. H. (2018). Server consolidation techniques in virtualized data centers of cloud environments: A systematic literature review. Software: Practice and Experience, 48(9):1688–1726. (Cited on pages 34 and 60.)

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2012). Foundations of machine learning. MIT press. (Cited on pages 47 and 50.)

Müller, A. C., Guido, S., et al. (2016a). Introduction to machine learning with python: a guide for data scientists. " O'Reilly Media, Inc.". (Cited on page 99.)

Müller, H. and Bosse, S. (2016). Multidimensional workload consolidation for enterprise application service providers. In Proceedings of the 26nd Americas Conference on Information Systems. (Cited on pages 11, 12, 70, and 77.)

Müller, H., Bosse, S., Pohl, M., and Turowski, K. (2017a). Capacity planning as a service for enterprise standard software. In Business Informatics (CBI), 2017 IEEE 19th Conference on Business Informatics, pages 167–175. (Cited on page 12.)

Müller, H., Bosse, S., and Turowski, K. (2016b). Optimizing server consolidation for enterprise application service providers. In Proceedings of the 2016 Pacific Asia Conference on Information Systems. (Cited on pages ix, 2, 3, 5, 11, 12, 28, 30, and 33.)

Müller, H., Bosse, S., and Turowski, K. (2017b). Capacity management as a service for enterprise standard software. Complex Systems Informatics and Modeling Quarterly, pages 1–21. (Cited on pages 12, 48, 49, and 80.)

Müller, H., Bosse, S., Wirth, M., and Turowski, K. (2017c). Collaborative software performance engineering for enterprise applications. In Proceedings of the 50th Hawaii International Conference on System Sciences. (Cited on pages 12, 48, 49, and 100.)

Müller, H., Görling, C., Hintsch, J., Splieth, M., Starke, S., and Turowski, K. (2016c). Monitoring energy consumption on the service level. In Proceedings of the 6th International Conference on Cloud Computing and Services Science-Volume 1 and 2, pages 215–222. (Cited on page 12.)

Müller, H., Prusch, A., and Agel, S. (2014). Hipas: High performance adaptive schema migration - evaluation of a self-optimizing database migration. In DEPEND 2014 : The Seventh International Conference on Dependability. (Cited on page 13.)

Müller, H., Prusch, A., and Agel, S. (2015a). Hipas: High performance adaptive schema migration - development and evaluation of self-adaptive software for database migrations. International Journal On Advances in Software, 8:262–275. (Cited on page 13.)

Müller, H., Splieth, M., Bosse, S., and Turowski, K. (2015b). Self-configuring data imports for sap hana cloud environments. In Proceedings of the Second HPI Cloud Symposium" Operating the Cloud" 2014, page 45. (Cited on page 11.)

Müller, H. and Turowski, K. (2015). Big data on performance logs - a collaborative monitoring cloud for erp systems. In Proceedings on the International Conference on Internet Computing (ICOMP), page 75. (Cited on pages 7, 11, and 15.)

Ng, F., Nag, S., ling Lam, L., Dharmasthira, Y., Eschinger, C., Anderson, R. P., Torn-

bohm, C., Roth, C., Tramacere, G., Blackmore, D., Wurster, L. F., Contu, R., Biscotti, F., Pang, C., Singh, T., Swinehart, H. H., Montgomery, N., Dominy, M., Petri, G., Kandaswamy, R., Palanca, T., Hare, J., Woodward, A., and Corriveau, J. (2018). Forecast: Public cloud services, worldwide, 2016-2022, 1q18 update. Technical Report G00343436, Gartner, Inc. (Cited on pages 27, 42, and 65.)

Niehorster, O., Krieger, A., Simon, J., and Brinkmann, A. (2011). Autonomic resource management with support vector machines. In 2011 12th IEEE/ACM International Conference on Grid Computing (GRID), pages 157–164. (Cited on pages 30, 47, 49, and 99.)

Orgerie, A.-C., De Assuncao, M. D., and Lefevre, L. (2014). A Survey on Techniques for Improving the Energy Efficiency of Large Scale Distributed Systems. ACM Computing Surveys, 46(4):1–35. (Cited on pages 2 and 32.)

Page, A. J. and Naughton, T. J. (2005). Dynamic task scheduling using genetic algorithms for heterogeneous distributed computing. In Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International, pages 8–pp. (Cited on page 36.)

Pelley, S., Meisner, D., Wenisch, T. F., and VanGilder, J. W. (2009). Understanding and abstracting total data center power. In Workshop on Energy-Efficient Design. (Cited on page 28.)

Petrucci, V., Carrera, E. V., Loques, O., Leite, J. C. B., and Mossé, D. (2011). Optimized management of power and performance for virtualized heterogeneous server clusters. In 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), Newport Beach, CA, USA, pages 23–32. (Cited on pages 29, 69, and 71.)

Pinheiro, E., Bianchini, R., Carrera, E. V., and Heath, T. (2001). Load balancing and unbalancing for power and performance in cluster-based systems. Technical report, Rutgers University. (Cited on pages 60, 61, 64, and 95.)

Pires, F. L. and Barán, B. (2013). Multi-objective virtual machine placement with service level agreement: A memetic algorithm approach. In Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing, pages 203–210. (Cited on pages 3, 32, 39, 40, 41, 67, and 71.)

Pollock, N., Williams, R., and Procter, R. (2003). Fitting standard software packages to non-standard organizations: the ́biographyóf an enterprise-wide system. Technology Analysis & Strategic Management, 15(3):317–332. (Cited on pages 4, 6, and 16.)

Pries-Heje, L. and Dittrich, Y. (2009). Erp implementation as design: Looking at participatory design for means to facilitate knowledge integration. Scandinavian Journal of Information Systems, 21(2):4. (Cited on page 6.)

Rabl, T., Gómez-Villamor, S., Sadoghi, M., Muntés-Mulero, V., Jacobsen, H.-A., and Mankovskii, S. (2012). Solving big data challenges for enterprise application performance management. Proceedings of the VLDB Endowment, 5(12):1724–1735. (Cited on page 4.)

Rathfelder, C., Klatt, B., Sachs, K., and Kounev, S. (2014). Modeling event-based communication in component-based software architectures for performance predictions. Software & Systems Modeling, 13(4):1291–1317. (Cited on page 79.)

Rockafellar, R. T. (1993). Lagrange multipliers and optimality. SIAM review, 35(2):183–238. (Cited on page 55.)

Rolia, J., Andrzejak, A., and Arlitt, M. (2003). Automating enterprise application placement in resource utilities. In Brunner, M. and Keller, A. (eds.), 14th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM), pages 118–129. (Cited on pages 36, 62, 71, and 91.)

Rolia, J., Cherkasova, L., Arlitt, M., and Andrzejak, A. (2005). A capacity management service for resource pools. In 5th International Workshop on Software and Performance (WOSP), pages 229–237. (Cited on pages 61, 62, 70, 71, and 94.)

Russell, S. J. and Norvig, P. (2010). Artificial intelligence: a modern approach. Malaysia; Pearson Education Limited,. (Cited on page 94.)

Rüßmann, M., Lorenz, M., Gerbert, P., Waldner, M., Justus, J., Engel, P., and Harnisch, M. (2015). Industry 4.0: The future of productivity and growth in manufacturing industries. Boston Consulting Group, 9. (Cited on page 1.)

Sahoo, J., Mohapatra, S., and Lath, R. (2010). Virtualization: A survey on concepts, taxonomy and associated security issues. In Computer and Network Technology (ICCNT), 2010 Second International Conference on, pages 222–226. (Cited on page 30.)

SAP (2012). Adaptive computing. URL http://scn.sap.com/docs/DOC-8646. Last accessed: January 25, 2019. (Cited on page 31.)

SAP (2017). Measuring in saps. URL https://www.sap.com/about/benchmark/measuring.html. Last accessed: January 14, 2019. (Cited on page 25.)

SAP (2018a). Business transaction. URL https://help.sap.com/saphelp_crm70/helpdata/en/4a/f39bf2f3c56530e10000000a42189b/frameset.htm. (Cited on page 7.)

SAP (2018b). Sap hana r integration guide v. 1.2. Technical report, SAP. (Cited on page 139.)

SAP (2019a). Sales and distribution (sd and sd-parallel). URL http://global.sap.com/

campaigns/benchmark/appbm_sd.epx. Last accessed: February 6, 2019. (Cited on pages 44 and 49.)

SAP (2019b). Sap netweaver. URL https://www.sap.com/products/netweaver-platform.html. Last accessed: January 25, 2019. (Cited on page 31.)

Sapru, V., Reddy, K., and Sivaselvan, B. (2010). Time table scheduling using genetic algorithms employing guided mutation. In Computational Intelligence and Computing Research (ICCIC), 2010 IEEE International Conference on, pages 1–4. (Cited on page 36.)

Schapire, R. E. (2013). Explaining adaboost. In Empirical inference, pages 37–52. Springer. (Cited on page 56.)

Setzer, T. and Stage, A. (2010). Decision support for virtual machine reassignments in enterprise data centers. In IEEE/IFIP Network Operations and Management Symposium (NOMS) Workshops, Osaka, Japan. (Cited on page 70.)

Shaeffer, D. L. (1980). A model evaluation methodology applicable to environmental assessment models. Ecological Modelling, 8:275–295. (Cited on page 57.)

Shalizi, C. (2006). Lecture 10: Regression trees. URL http://www.stat.cmu.edu/~cshalizi/350-2006/lecture-10.pdf. Last accessed: February 15, 2019. (Cited on pages ix, 50, 52, and 53.)

Shaw, P. (2004). A constraint for bin packing. In International conference on principles and practice of constraint programming, pages 648–662. (Cited on pages 3 and 39.)

Shehabi, A., Smith, S., Sartor, D., Brown, R., Herrlin, M., Koomey, J., Masanet, E., Horner, N., Azevedo, I., and Lintner, W. (2016). United states data center energy usage report. URL https://escholarship.org/uc/item/84p772fc. Last accessed: January 14, 2019. (Cited on pages 3, 28, and 29.)

Shi, L., Butler, B., Wang, R., Botvich, D., and Jennings, B. (2012). Optimal placement of virtual machines with different placement constraints in iaas clouds. In Symposium on ICT and Energy Efficiency and Workshop on Information Theory and Security (CIICT 2012). (Cited on page 41.)

Skene, J., Raimondi, F., and Emmerich, W. (2010). Service-level agreements for electronic services. IEEE Transactions on Software Engineering, 36(2):288–304. (Cited on page 20.)

Smith, C. U. (1981). Increasing information systems productivity by software performance engineering. In Seventh International Computer Measurement Group Conference, New Orleans, LA, USA, December 1-4, 1981, Proceedings, pages 5–14. (Cited on page 47.)

Soltani, R. (2014). Reliability optimization of binary state non-repairable systems: A state of the art survey. International Journal of Industrial Engineering Computations, 5(3):339–364. (Cited on page 34.)

Somers, T. M. and Nelson, K. (2001). The impact of critical success factors across the stages of enterprise resource planning implementations. In System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on, pages 10–pp. (Cited on pages 4, 6, 16, and 17.)

Sonnenberg, C. and vom Brocke, J. (2012). Evaluations in the science of the artificial–reconsidering the build-evaluate pattern in design science research. In International Conference on Design Science Research in Information Systems, pages 381–397. (Cited on pages 9, 13, 113, 114, and 130.)

Speitkamp, B. and Bichler, M. (2010). A mathematical programming approach for server consolidation problems in virtualized data centers. IEEE Transactions on Services Computing, 3:266–278. (Cited on pages 2, 3, 22, 29, 33, 39, 40, 41, 65, 70, 71, 78, 79, 81, and 85.)

Splieth, M., Bosse, S., Schulz, C., and Turowski, K. (2015). Analyzing the effects of load distribution algorithms on energy consumption of servers in cloud data centers. In Proceedings of the 12th International Conference on Wirtschaftsinformatik (WI). (Cited on pages 2, 28, and 33.)

Stillwell, M., Schanzenbach, D., Vivien, F., and Casanova, H. (2010). Resource allocation algorithms for virtualized service hosting platforms. Journal of Parallel and Distributed Computing, 70:962–974. (Cited on pages 3, 34, 35, 36, 38, 65, 68, 69, 71, 90, 91, and 92.)

Sun, M., Gu, W., Zhang, X., Shi, H., and Zhang, W. (2013). A matrix transformation algorithm for virtual machine placement in cloud. In 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, pages 1778–1783. (Cited on pages 29, 33, and 66.)

Susanta, N. and Chiueh, T.-c. (2005). A survey on virtualization technologies. Rpe Report, 142. (Cited on page 30.)

Suykens, J. A. and Vandewalle, J. (1999). Least squares support vector machine classifiers. Neural processing letters, 9(3):293–300. (Cited on page 55.)

Sydor, M. J., Sleeth, K., Toigo, J., Yourdon, E., Donaldson, S. E., Siegel, S. G., and Donaldson, G. (2010). Apm best practices: Realizing application performance management. Apress. (Cited on page 4.)

Tertilt, D. and Krcmar, H. (2011). Generic performance prediction for erp and soa applications. In ECIS. (Cited on page 47.)

Tudenhöfner, E. (2011). Integration of performance management into the application lifecycle. diplom.de. (Cited on pages 2, 4, 15, and 47.)

Van, H. N., Tran, F. D., and Menaud, J.-M. (2009). Sla-aware virtual resource management for cloud infrastructures. In 9th IEEE International Conference on Computer and Information Technology (CIT'09), pages 1–8. (Cited on pages 28, 64, and 69.)

Varasteh, A. and Goudarzi, M. (2017). Server consolidation techniques in virtualized data centers: A survey. IEEE Systems Journal, 11(2):772–783. (Cited on pages 60, 70, 71, and 89.)

Venkataraman, S., Yang, Z., Franklin, M., Recht, B., and Stoica, I. (2016). Ernest: efficient performance prediction for large-scale advanced analytics. In 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16), pages 363–378. (Cited on pages 47, 48, 49, and 99.)

Walls, J. G., Widmeyer, G. R., and El Sawy, O. A. (1992). Building an information system design theory for vigilant eis. Information systems research, 3(1):36–59. (Cited on page 5.)

Walter, S. M., Böhmann, T., and Krcmar, H. (2007). Industrialisierung der it - grundlagen, merkmale und ausprägungen eines trends. HMD Praxis der Wirtschaftsinformatik, 44(4):6–16. (Cited on page 1.)

Westermann, D., Happe, J., Hauck, M., and Heupel, C. (2010). The performance cockpit approach: A framework for systematic performance evaluations. In Software Engineering and Advanced Applications (SEAA), 2010 36th EUROMICRO Conference on, pages 31–38. (Cited on pages 45 and 47.)

Whitley, L. D., Starkweather, T., and Fuquay, D. (1989). Scheduling problems and traveling salesmen: The genetic edge recombination operator. In ICGA, pages 133–40. (Cited on page 93.)

Wilhelm, K. (2001). Capacity planning for sap-concepts and tools for performance monitoring and modelling. CMG Journal of Computer Resource Management. (Cited on pages 16, 95, and 97.)

Wilhelm, K. (2003). Messung und modellierung von sap r/3-und storage-systemen für die kapazitätsplanung. PhD thesis, University of Duisburg-Essen. (Cited on pages 46, 49, and 95.)

Williams, L. G. and Smith, C. U. (1998). Performance evaluation of software architectures. In Proceedings of the 1st international workshop on Software and performance, pages 164–177. (Cited on page 44.)

Wirth, M. (2015). Entwicklung von modellen zur antwortzeitvorhersage von transaktionen betrieblicher standardsoftware am beispiel von sap erp. Master's thesis, Fakultät für Informatik, Otto-von-Guericke-University Magdeburg. (Cited on pages 48, 97, and 100.)

Wolf, C. and Halter, E. M. (2005). Virtualization: From the desktop to the enterprise. Apress. (Cited on page 30.)

Wolpert, D. H., Macready, W. G., et al. (1997). No free lunch theorems for optimization. IEEE transactions on evolutionary computation, 1(1):67–82. (Cited on page 125.)

Wood, T., Shenoy, P. J., Venkataramani, A., Yousif, M. S., et al. (2007). Black-box and gray-box strategies for virtual machine migration. In NSDI, pages 17–17. (Cited on pages 25, 62, and 71.)

Xu, J. and Fortes, J. A. B. (2010). Multi-objective virtual machine placement in virtualized data center environments. In IEEE/ACM International Conferenc on Cyber, Physical and Social Computing (CPSCom), Hangzhou, China. (Cited on pages 29, 36, 64, 68, and 91.)

Yoo, W., Larson, K., Baugh, L., Kim, S., and Campbell, R. H. (2012). Adp: automated diagnosis of performance pathologies using hardware events. ACM SIGMETRICS Performance Evaluation Review, 40(1):283–294. (Cited on pages 44, 48, and 49.)

Yusoh, Z. I. M. and Tang, M. (2012). A penalty-based grouping genetic algorithm for multiple composite saas components clustering in cloud. In Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on, pages 1396–1401. (Cited on pages 40, 41, and 91.)

Zhang, Q., Cherkasova, L., Mathews, G., Greene, W., and Smirni, E. (2007). R-capriccio: A capacity planning and anomaly detection tool for enterprise services with live workloads. In ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing, pages 244–265. (Cited on page 19.)

Ehrenerklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; verwendete fremde und eigene Quellen sind als solche kenntlich gemacht. Insbesondere habe ich nicht die Hilfe eines kommerziellen Promotionsberaters in Anspruch genommen. Dritte haben von mir weder unmittelbar noch mittelbar geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.

Ich habe insbesondere nicht wissentlich:

- Ergebnisse erfunden oder widersprüchliche Ergebnisse verschwiegen,
- statistische Verfahren absichtlich missbraucht, um Daten in ungerechtfertigter Weise zu interpretieren,
- fremde Ergebnisse oder Veröffentlichungen plagiiert,
- fremde Forschungsergebnisse verzerrt wiedergegeben.

Mir ist bekannt, dass Verstöße gegen das Urheberrecht Unterlassungs- und Schadensersatzansprüche des Urhebers sowie eine strafrechtliche Ahndung durch die Strafverfolgungsbehörden begründen kann. Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form als Dissertation eingereicht und ist als Ganzes auch noch nicht veröffentlicht.

Magdeburg, den 17. Mai 2019

. . . . . .
Hendrik Müller, M. Sc.