Data Governance and Semantic Recommendation Algorithms for Cloud Platform Selection

Stefan Kolb and Guido Wirtz
Distributed Systems Group
University of Bamberg
Bamberg, Germany
{*stefan.kolb, guido.wirtz*}*@uni-bamberg.de*

***Abstract*—Platform as a Service is the major productivity enabler in the cloud computing stack. By providing managed and highly automated application environments, it enhances developer productivity and reduces developer operations and maintenance efforts. The market, however, is fast-changing and offerings are differing conceptually as well as in their supported technological ecosystem. Therefore, provider selection is an important but currently not well supported step for companies trying to benefit from the technology. Influenced by the diversity of service offerings and the absence of applied standards this is a tedious task, especially for ensuring application portability. In this paper, we present a multi-criteria selection approach for cloud platforms based on a field-tested ontology and a comprehensive data set. The methodology is enhanced by semantic algorithms and mappings to reduce hidden query and data biases. This allows not only the exact matching of requirements but also the evaluation of possible alternatives that can be adapted to fit the defined requirements. We validate our approach by contrasting real user queries against the results of our semantically enhanced algorithms.**

Keywords*—*Cloud Computing, Platform as a Service, PaaS, Selection, Decision Making, Semantics, Recommender, Portability

I. Introduction

Platform as a Service (PaaS) is a major technology to improve development productivity for today's agile development cycles. The managed and highly automated application environments free developers from configuring servers and reduce developer operations and maintenance efforts. As a result, developers can focus on the application development, generating the actual business value. The PaaS market itself, however, is fragmented and offerings are differing conceptually as well as in their supported technological ecosystem [1]. Therefore, provider selection is an important but currently not well supported task for companies trying to benefit from the technology. Besides making an informed choice, despite the variety of offerings, a main issue within this task is to avoid a potential vendor lock-in and retain future options for application portability [2]–[4]. In contrast to Infrastructure as a Service (IaaS), decisions on appropriate PaaS providers typically involve more criteria due to their diverse ecosystems. As of now, there are no widely applied standards in the world of PaaS which stresses another approach based on technological components and capabilities to compare offerings [1]. Until recently, the vendors' documentation and advertisements were the only source of data to answer these questions. Not only because such unstructured information may be updated without prior notice, but also since the whole process involves a lot of manual tasks that are costly, the need for a consolidated repository evolved [1], [5], [6]. Nonetheless, existing works targeting cloud provider selection regularly lack a decent data set in terms of amount, actuality, and quality. Moreover, the approaches often neglect important real-world problems and are based on an optimistic view on the quality of the data and the users' selection queries. As we will show, these assumptions are often not sufficient for the practical applicability of provider selection and ignore an evident problem of data and query biases while sacrificing results. Closely associated, most approaches only allow an exact matching of the users' queries with the data which can lead to a substantial amount of unsatisfied queries. However, in most scenarios there is a chance that partial matches might still be able to fit the user's needs constrained by a set of semantic rules for the applied domain.

Whereas feasible algorithms for selection are discussed fairly often, data and query problems and semantic knowledge lack appropriate consideration. The presented work aims to improve on these shortcomings in general and as an application of the problem in the context of cloud platform selection. Hence, the focus of this paper is on cloud properties and semantic matching for cloud platforms rather than on generic decision algorithms that are discussed sufficiently in related works [7], [8]. We stress that existing works on cloud selection are of limited value in practice because they are missing additional validation and semantic enhancements to improve selection accuracy. To that end, we will validate our hypotheses in practice via real-world data from the leading cloud platform knowledge base *PaaSfinder*[1]. Our main research questions are:

RQ 1: How to find matching cloud platforms for particular application and user requirements?

RQ 2: Are there any issues with the accuracy and satisfaction of user queries caused by data and query biases?

RQ 3: How to semantically enhance matching algorithms to find and rank cloud platforms that only partially match the defined requirements?

RQ 4: Do these algorithms improve the satisfaction and accuracy of real user queries?

To elaborate on the given questions, the paper proceeds as follows: In Section II, we describe important fundamentals for our selection and portability approach targeting *RQ 1.* Next, we evaluate real user queries to answer *RQ 2* before we present our

[1]See https://PaaSfinder.org

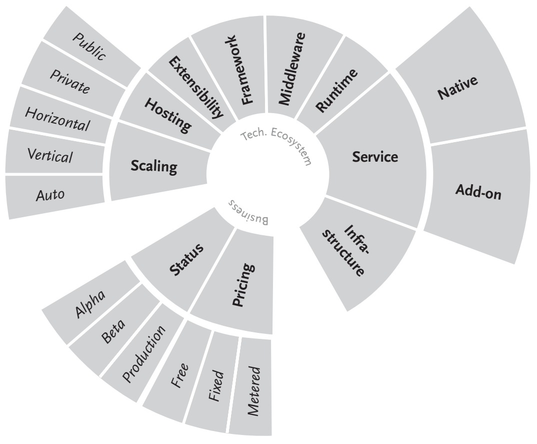

Fig. 1: Platform as a Service Taxonomy (cond.) [1]

methodologies for data governance and semantically enhanced selection algorithms (*RQ 3&4*) to account for data and query biases in Sections III and IV. In Section V, we contrast our approach with existing related work. Section VI discusses limitations and future work. Finally, Section VII summarizes the contributions of the paper.

II. Cloud Platform Selection

A. PaaS Taxonomy

Due to conceptual differences, each cloud service model (IaaS, PaaS, SaaS) needs to be treated separately in terms of comparison, selection or portability [9]. Whereas the entities and interfaces of IaaS systems like compute, network, and storage [10] are widely agreed upon, those of PaaS offerings are less well described by current standards and lacking a common terminology or model [5], [11]–[13]. Taxonomies and semantic technology is regarded as one of the most efficient solutions for the classification, normalization, and connection of domain knowledge [5]. To be able to supply a central marketplace and compare existing PaaS providers, we first needed to agree on a PaaS taxonomy to index offerings. Our PaaS taxonomy [1] is based on extensive literature reviews, analysis of the state of the art, and longstanding experience with cloud platforms and their evolution. Moreover, the accompanying data set of currently 74 active PaaS vendors is the most recent and comprehensive available to date. The referenced taxonomy and its attributes are the foundation for the structured data set of the knowledge base and the presented selection algorithms and semantics. As every knowledge base is only an abstraction of reality and limited to a specific point of view, the set of attributes of a derived taxonomy differs. In our case, the selection and definition of relevant attributes is defined by a notion of application portability based on the technological ecosystem of the providers. Thus, the model's attributes are focused on the available technological ecosystem that is vital for assessing application portability between vendors [1]. Other properties like business-related QoS or cost details are omitted to some extent. Nevertheless, the specification is extensible for any kind of qualitative or quantitative attributes. Figure 1 shows an excerpt of the taxonomy's properties that are available through the filtering interface to users conducting a cloud provider selection. For a comprehensive explanation of the model and its attributes

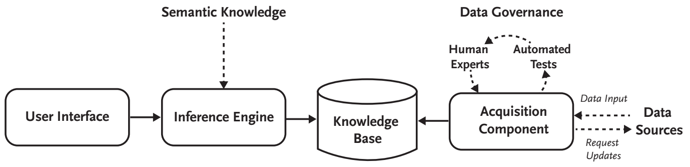

Fig. 2: Decision Support System Architecture

please see [1] and the project repository[2]. Technically, the data is structured and encoded as JSON profiles so that the knowledge representation is applicable for both, machines and humans. The following section gives an insight how the notion that lead to the selection of attributes was derived and what it means.

B. Ecosystem Portability

One of the major obstacles when working with software systems is how to prevent or tackle application portability threats. An obvious answer to this is often standardization. Naturally, several standardization organizations have addressed cloud standards. Whereas some of them have gained traction, e.g., OVF [14], most initiatives remain disregarded by practice. Moreover, only very few of them consider the PaaS model as their main objective rather than IaaS. Examples in the context of application portability as single unit of delivery include TOSCA [15] or the Open Container Initiative[3]. Yet, we can see that vendors already have competing ideas and approaches in this area. In the past, software standards have failed for achieving portability in various ways [16]–[18]. For the cloud, especially the lack of acceptance by industry leaders prevents adoption and the market is also still too fragmented and evolving at the moment.

For that reason, we pursue a no-standards approach for application portability with no intermediaries and instant applicability. Commonly, portability is based on a set of attributes that bear on the ability of software to be transferred from one environment to another [19]. More specifically, portability can be based on application dependencies which means relying on native support and open technologies. In our data, we see consensus in an array of dependencies that are supplied and used for typical application development. Naturally, vendors want to attract as many customers as possible by supporting their development needs which is why their ecosystems intersect. Hence, if all required technological components and capabilities are supported by a platform, we should be able to run our application with little to no additional adaption effort [20]. Next to the components of the software, the characteristics of the service provider become an important factor when selecting a software service [21]. For instance, the location of the provider's data center and the implemented privacy policy are often important nonfunctional criteria [22]. A benefit of our approach is that it can deal with both of the described requirements. Whereas this principle weakens the typical *write once, run anywhere* cross-platform benefits of standards, it has a wider range of applications. The feasibility of the described approach was validated by two case studies including an extensive real-world application migration [1], [23], [24].

[2]See https://github.com/stefan-kolb/paas-profiles
[3]See https://www.opencontainers.org

TABLE I: Statistics for User Interactions with Exact Matching Algorithm

	Exact matching
Number of satisfied queries	5836 (73.78 %)
Number of unsatisfied queries	2074 (26.22 %)
$\sum$ Total queries	7910

C. Decision Support System

Figure 2 shows the general architecture of the Decision Support System (DSS) *PaaSfinder*. The central part of the system is of course the knowledge base with the PaaS provider data. The data is entered and updated via the acquisition component via different data sources and stakeholders. Several quality control stages are implemented along this way which are described in the following to ensure data quality. The interface to the inference component of the knowledge base supports both human and machine decision makers, which we believe is essential for such a decision that is both automatable and technically influenced, but finally driven by human decision makers.

In the introduction, we assert that the quality of the data and the users' queries have a possible negative influence on the selection result leading to fewer appropriate results or no results. To elaborate this hypothesis, we state *RQ 2: Are there any issues with the accuracy and satisfaction of user queries caused by data and query biases?* To evaluate this question, Table I shows aggregated statistics of user queries recorded by *PaaSfinder*. The queries were conducted by real users with an interface for exact matching as described in Section IV-A. A total of 7910 interactions from 4200 users were recorded in between 8/2/2016 and 01/18/2017[4]. We define *satisfied* queries as queries that returned at least one PaaS provider that fits the requirements specified by the user as result. This does not necessarily mean that the user found the appropriate provider for his needs, but the query itself is satisfied. Consequently, all queries that returned no result are categorized as *unsatisfied*. As we can see, 73.78 % of the user queries provided at least one result. Then again, this means that more than a quarter of them did not satisfy the user's query. By carefully examining the set of unsatisfied queries, we realized that there is a larger set of queries that could have been satisfied with additional expert knowledge. In this context, an expert is a person with special skills or experience in the particular area, who is widely recognized as a reliable source of knowledge in that area [25]. But such experts are costly and therefore it would be beneficial to externalize parts of such knowledge once and reapply it automatically to different queries on the system. But were do the problems come from that prevent a satisfying query result and can they be prevented by semantically enhanced selection algorithms? To answer this question, we analyzed the queries, data, and the results and categorized what led to the unsatisfied query result. Thereby, we identified two main categories of problems that caused the mismatches: query and data problems. Both types influence each other to create unwanted effects on the query results, e.g, no results. Figure 3 summarizes these potential problems for queries on knowledge bases intertwined with our approach to include semantic knowledge to account for these issues. As we see, these problems are often neglected in related work on cloud provider selection and diminish the applicability of the research results for practice.

[4]Users are identified based on their IP address. Technically induced duplicate requests were removed from the data. The anonymized raw data can be accessed at https://github.com/stefan-kolb/paas-profiles/releases/tag/cloud17

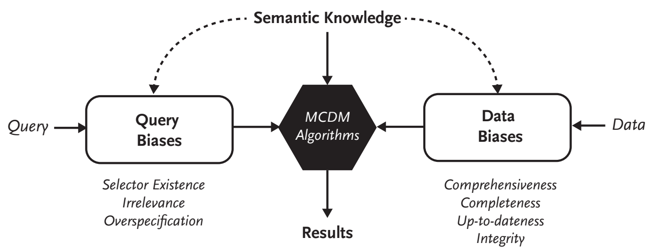

Fig. 3: Selection Problems Impacting MCDM Scenarios

III. Data Biases and Data Governance

A comprehensive, correct, and consistent knowledge base is the foundation of a methodical comparison and selection of cloud platforms. So an important step during the course of any MCDM approach is to continuously secure the data quality through appropriate data governance. This must be enforced by a carefully crafted data model at design time and quality assurance inside the acquisition component at runtime.

Several of the data problems depicted in Figure 3 that occur at runtime already manifest themselves during design time of the data model. As we discussed in Section II-A, the decision problem for the selection plays an important role for the scope of the model's attributes. This point of view also defines the relevant data for the knowledge base. For cloud platforms, we realized that there exists a conflict between the aims of knowledge bases and the practical applicability in related work. Not all properties that seem relevant for a particular problem can be fulfilled or are available for a majority of entities in the real-world, e.g., SLAs or CPU power values. Hence, typical problems that occur at runtime are missing or incomplete data (*comprehensiveness, completeness*). Sometimes, possible candidates are not available inside the recent data set (*comprehensiveness*) due to the fast moving business or specific information is not listed due to the amount and complexity of the model's properties (*completeness*). Often, data actuality is limited as providers are frequently changing their offerings but corresponding data is not available in a structured form and must be manually updated by humans (*up-to-dateness*). Also, several threats to the *integrity* of the data due to consistency problems and errors are common. Not all of the domain-specific data consistency rules can be defined inside the data model, so missing information on derived or related data entries such as Cloud Foundry-based platforms and different naming terminologies or faulty values cannot be completely eliminated [6], [13]. Also, there exist several threats that exacerbate the identified data biases. First of all, stale data and infrequent updates are problems of knowledge bases that often occur. Next, we experienced data biases due to provider interests and misinformation that were caused by different knowledge levels and opinions of data suppliers.

Due to the presented problems, special care to the data governance needs to be taken in knowledge-based systems.

In our case, one might think that an automatic information gathering process of the platforms' specifications is possible. However, there is virtually no data available in a standardized and structured form that would allow us to crawl information automatically. Instead, data comes from heterogeneous sources and most often in natural language. Examples are the provider websites, user documentations, changelogs, and only sometimes APIs. In such cases, the data is typically gathered and entered through the acquisition component by experts or knowledge engineers. Due to the amount, diversity, and continuous changes to the PaaS provider market and data this is not feasible. Therefore, we use a more open process with additional knowledge providers to enhance data completeness and actuality. Stakeholders are domain experts, vendors, consultants, end users, and automatic crawlers. Additionally, a feedback loop back to the knowledge engineers to trigger a revalidation or an update of the data after a certain threshold is desirable to avoid stale data. As the various knowledge suppliers have different levels of expertise and intentions, we needed to implement several techniques and barriers inside the acquisition component to ensure good data quality (see Figure 2). First and foremost, data structure requirements and limitations for available properties and values are enforced by our generic PaaS model. Additionally, we implemented a large set of automated semantic rules for the data inside the acquisition component that cannot be checked by model constraints, to avoid, e.g., duplicates, intersecting concepts [13], [26] and ensure completeness, consistency of concepts and values where possible. Nevertheless, a second human quality control stage must be passed, if the data passes the tests, before the data is merged. Solely ensuring that the structural and semantical integrity of the data is fine via automatic tests is not enough because there still is the problem that wrong information might be given unintentionally or intentionally which can only be reasonably validated by human experts. As an example, we experienced that IaaS vendors which did not provide specific PaaS features tried to add their offering to the knowledge base for marketing reasons.

With all of the presented data governance measures, we prevent numerous typically unhandled threats to data quality up front. However, since absolute completeness and consistency of the data is hard to achieve in reality, this must be targeted by additional semantic enhancements to the query as shown in the following section.

IV. Query Biases and Semantic Query Enhancements

Query problems are caused by the user or the machine that sends a query to the knowledge base. In some cases, query data may be incomplete, inaccurate, or simply irrelevant to the problem that is being investigated [25]. This includes nonexistent query attributes or values that must be prevented by structural query validation to avoid unsatisfied queries (*selector existence*). In our case, a query is validated by our feature model and its validations by translating it into a virtual provider model. Other effects are harder to detect and handle. For example, if the user does not ask the right question for his problem, e.g., a user is looking for hosting in North America but does not include the option for a private hosting in his query. If the model does not cover the relevant properties of the decision problem, the query cannot be answered with an appropriate result. Also, the more detailed a query is, the more comprehensive the data set needs to be. As a state of total information is not achievable in reality, we need to assume the imperfectness of the data set. A too specific query in connection with an imperfect data set can lead to an unsatisfied query whereas the actual requirements might still be satisfiable in reality. In general, there is a correlation between very specific queries and a no result. Our data shows that unsatisfied queries have an average of 5.38 query keys whereas satisfied queries have only 2.72 keys per query selector. The typical search behavior applied by users can often be described by a back and forth of specialization and generalization of the search query. One of our ideas to tackle this problem is to automatically assist the user on his way in this process via semantic knowledge to make it feasible to relax possibly overspecified queries and show the effects of query relaxations to the user.

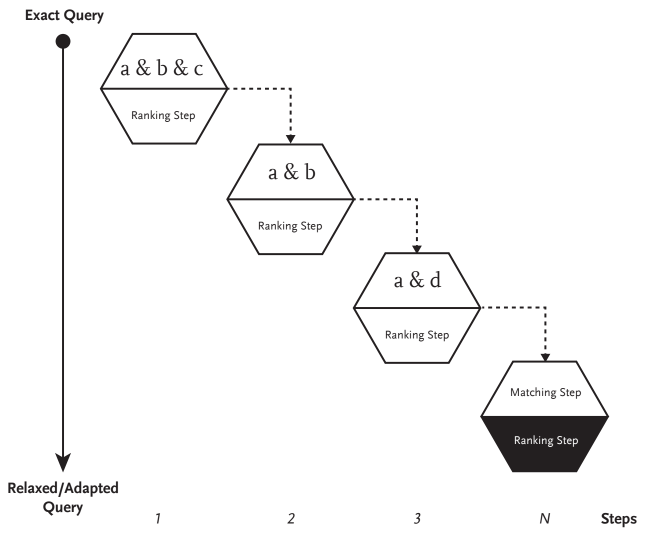

Fig. 4: Multi-step Selection Process

Therefore, to mitigate the observed effects, in addition to the presented enhancements to the knowledge acquisition, we propose an enhanced multi-step selection process (see Figure 4). Overall, we try to answer a typical MCDM question: 'given a set of alternatives and a set of decision criteria, then what [are] the best alternative[s]?' [8, p. XXV] First, we try to fully satisfy the user query. In case we could not find suitable vendors or to provide possible alternatives, we relax and adapt the query based on externalized semantic knowledge. In the following, we first discuss the exact matching algorithm and afterwards our semantic algorithms that allow such a partial matching (*RQ3*).

A. Exact Matching Step

Ultimately, a query result that fully satisfies the user's demands is what we always should aim for. This is especially necessary for automatic application migration scenarios, where we cannot relax the query and risk incompatibilities that would potentially break the portability and the process by forcing adaptations to the application. In fact, the exact matching step can be broken down into two steps again, a filtering and a ranking step. As our main focus for the selection and the properties of our PaaS model is on application portability, most of the criteria are must-haves. This means if any of those criteria cannot be fulfilled by a candidate provider, it must be excluded from the result set (*filtering step*). Most of the popular MCDM algorithms do not handle such must-have criteria well but optimize a target function. Such a step can be best applied after the initial filtering of portable candidates to rank the results based on non must-have criteria, e.g., pricing or uptime, with established MCDM methods (*ranking step*). To this end, several well known algorithms such as the analytic hierarchy process (AHP) [27], outranking [28] or the weighted sum model [29] can be used. By now, inside the filtering step, all requirements are equally important for the selection algorithm. For our case, this is reasonable as most requirements are relevant for application portability and therefore must-haves. Also, user-defined preferences require more knowledge from the user what implications this has on the result set and wrong configurations might have unwanted side effects on the results. Therefore, in this scenario we follow a different approach and try to make it as easy as possible for the user to define his must-haves and let the algorithms do the hard work. However, we could add this functionality later to feed the algorithms with additional information what can safely be relaxed to expand the result set.

B. Semantic Matching Steps

As shown, there are data and query problems that cause unsatisfied queries or diminish the amount of possible alternatives that are displayed. Therefore, as a second step of our selection approach we introduce a partial matching stage (*RQ3*). It is useful for semi-automatic or manual application migration search or exploration with user interaction. This is reasonable as of now in most cases a fully automatic migration among cloud platforms will not be possible anyhow [23], [30]. Again, see Figure 4 for an illustration how we gradually relax and adapt the user's query to find new sets of (additional) appropriate selection results. In that case, we make use of certain rules and algorithms based on semantic information to alter the initial user query. These rules account for the data and query problems discussed before. Overall, there do exist some general algorithms that can be applied to nearly all knowledge bases of similar type than ours, e.g., similarity based on edit distances of query attributes. However, most of the general concepts need to be specifically tailored to the exact domain of the knowledge base [13], [25]. In the following, we present a set of rules for the aforementioned problems applied to the case of cloud platform selection. By using different domain-specific rules, several restrictions can be relaxed while still retaining a perfect fit or at least the possibility to find a fit. Even when there is no exact match, such a result can satisfy the user more than a no result. Some of the rules and algorithms are portability preserving while others imply the possibility that porting is possible but not guaranteed or involves more effort to achieve the same result. Therefore, these results must be marked with some kind of less appropriate indicator to the user. Of course this list of algorithms can be extended further. Here, we present a selection of our main findings inside the PaaSfinder system.

a) Generalization: What we learned from our work with the knowledge base is that the more specific the information is, the less likely it is that it is given or even up-to-date. Therefore, specific information that comes in a hierarchical context with other data can be more safely relaxed than generic information without influencing the selection result. Also, as we discussed before, users tend to overspecify their needs which also fosters the negative impact caused by missing or outdated data. Hierarchical relationships between data are often found in technological knowledge bases. For an example see equation 1.

$$Runtime \rightsquigarrow Framework \rightsquigarrow Version \tag{1}$$

A concrete rule from our data would be, Ruby $\rightsquigarrow$ Rails $\rightsquigarrow$ v5. In the course of the partial matching stage, we can relax queries that include very specific information, e.g, a framework version, so that they only request the framework and imply that the specific version may be supported. Even further, we can then relax the need for the framework to a simple requirement for the corresponding runtime and imply that this framework will be supported. Although this line of argument and relationship might sound ambitious, tests with our data show that it is actually often safe to assume.

b) Customization: Whereas today all cloud platforms come with a large ecosystem of technological assets, there is still room for customization and the addition of technologies by the user. This trend was initially introduced by the Buildpack[5] concept of Heroku, to allow the users to customize and add specific technologies to the preconfigured application environments just like they were used to with IaaS. This was necessary as more and more offerings evolved from specialized language-based PaaS to polyglot platforms supporting more and more runtimes and therefore the state space of ecosystems to be supported grew exponentially. Platforms that support such an extensibility concept are capable of running more technologies than officially supported by the provider. These concepts are typically best applied for the addition of other runtime languages and framework support. The selection algorithms can make use of that to include these vendors into the extended result set even when runtimes are not supported by the data. This especially helps with queries that look for a larger set of supported runtimes $(N(unsatisfied)_{runtimes>1} = 505)$ indicating the need for extensibility of the platform.

c) Compatibility: The concept of compatibility or replaceability can be applied in multiple ways inside the selection algorithms. In a classical sense, several technologies implement the same base technology or a standardized language, such as SQL, and are at least partially compatible and replacable, e.g., MySQL, PostgreSQL, and MariaDB. Often, providers also name their configuration of a system differently as it is tweaked differently, e.g., Pivotal GemFire which essentially is Apache Geode. Additionally, the ecosystem of cloud platforms can replace required native services through their third party add-on marketplaces. Often, it can be beneficial to use a specialized and managed add-on service for specific service dependencies rather than relying on a native service inside the platform.

d) Terminology: Even with a stringent data model and automated tests as presented in Section III, it cannot be guaranteed that every technology has the same consistent name throughout the data. Due to the large and diverse ecosystems,

[5]See https://devcenter.heroku.com/articles/buildpacks

TABLE II: Evaluation of Semantic Query Enhancements

Algorithm	Target	Query selector(s)	No. of unsatisfied queries	Pct. of unsatisfied queries	Avg. results $\tilde{x}$
Exact matching			2074	26.22 %	5
Generalization	completeness, up-to-dateness, overspecification	*frameworks*	1792 (-282)	22.65 % (-3.57 %)	8
Customization	completeness, up-to-dateness	*runtimes, frameworks*	1345 (-729)	17.00 % (-9.22 %)	16
Terminology	integrity	*middleware, frameworks, native services, add-ons*	2010 (-64)	25.41 % (-0.81 %)	5.5
Compatibility	integrity	*native services*	1992 (-82)	25.18 % (-1.04 %)	6
Portability	overspecification, irrelevance	*status*	1873 (-201)	23.68 % (-2.54 %)	7
		pricing	1686 (-388)	21.31 % (-4.91 %)	9

property values cannot be completely restricted and enumerated. In our case, this applies to frameworks, middleware products, services, and add-on names. Therefore, we also need to find equalities between them to correct inconsistencies and also add probable duplicates to the result sets. To that end, we can apply edit-distance based algorithms to identify similar technologies.

e) Portability: As we discussed before, every selection approach has its specific scope that manifests itself inside the knowledge base's model. Most of the times, several additional attributes are added to the model that contribute to the practical applicability of the selection and are often requested by users. Nevertheless, when using an excluding matching step, all criteria contribute equally to the candidate filtering regardless of their importance for the selection scope. To not require the users to apply this specific knowledge manually, the algorithms should be aware which attributes can be safely relaxed. In our case, we mainly focus on the portability of application dependencies. Therefore, it can be beneficial to relax criteria that are not immediately important for application portability, e.g., pricing.

C. Evaluation

In Section II-C, we already showed that there are issues with the accuracy and satisfaction of user queries caused by data and user query biases. To evaluate the effects of our proposed semantic enhancements, we need to reconsider *RQ 4*: *Do these algorithms improve the satisfaction and accuracy of real user queries?* For evaluation, we applied the suggested algorithms on the recorded user interactions[6] to evaluate how the user experience would have been altered with our proposed semantic query enhancements. Table II shows the results of both, the exact query matching and the semantically enhanced queries in relation to each other.

We can not only see that the number of unsatisfied queries decreases substantially due to the semantic enhancements but also that the number of average results (median) rises. Generally, more choices are not necessarily desirable as the decision problem for the user gets more complicated with a larger result set. Therefore, additional semantic steps are best applied to empty result sets or very few results, initially leaving the user with no real choice. Exactly these cases are frequent in our data ($N(results <= 1) = 2845, 35.97~\%$) which strengthens the necessity for the semantic matching steps. The *generalization* algorithm proves to be one of the most effective query enhancements. Apparently, the sizes of the platforms' technological ecosystems pose a problem for the completeness and up-to-dateness of the data set. The presented enhancements smooth out these inconsistencies while still preserving these frequently requested model attributes. Expectedly, the *customization* algorithm has the highest impact on the result set size as it appends every extensible platform to all runtime queries. Whereas it does retain application portability, due to its impact on the query, it should only be used sparingly and for the final stages of the selection process. The *terminology* results show that our data governance measures from Section III are working quite effectively, as we experienced only a few syntactic duplicates summing up for less than one percent of result set changes. For the *compatibility* mappings, we identified sets of 15 compatible native services. Even with this small set of relationships, we can improve the amount of satisfied queries by one percent. This could be further enhanced by identifying appropriate mappings to available third party add-on services. Last, we evaluated two examples for relaxing query attributes, i.e., status and pricing, that do not directly contribute to the *portability* of application dependencies. Both highly influence the user request but can be valuable to satisfy queries that need to focus strongly on application dependencies first. Overall, we can conclude that the more detailed queries a knowledge base allows, the more beneficial our semantic enhancements and matching stages will be.

D. Summary

Together with our work from [1], we proposed a system for selecting cloud platforms for particular application and user requirements (*RQ1*). By examining real user queries from the PaaS knowledge base *PaaSfinder*, we showed that there are issues with the accuracy and satisfaction of user queries (*RQ2*) in knowledge bases and suggested different semantic algorithms that account for the identified data and query biases that allow a partial matching of application requirements (*RQ3*). Our evaluations show that the proposed semantics enhance the user satisfaction and lead to a more accurate selection result (*RQ4*).

V. Related Work

Existing literature targeting cloud provider selection can be best distinguished based on the cloud model [9]. Essentially,

[6]The concrete implemented algorithms can be found at https://github.com/stefan-kolb/paas-profiles/releases/tag/cloud17

a majority of papers focus on IaaS, whereas little work has been conducted on PaaS cloud service selection [5]. In general, recent work on cloud service selection is primarily focused on rankings based on cost comparisons and other nonfunctional characteristics [5], [31]. Functional aspects are mostly only considered through virtual machine capabilities in the IaaS context. All of the referenced works validate their approaches on a very limited data set of cloud providers.

A. *Generic Approaches*

Despite the differences between cloud types, many existing approaches intend to suggest a generic solution for cloud services. However, this often limits their practical applicability due to the conceptual differences between cloud types and therefore small set of intersecting properties [9]. Wittern et al. [32] use feature models as a representation mechanism for requirements elicitation within a cloud service selection process. As a concrete instantiation of their generic selection approach, they present models for cloud storage selection. Sundareswaran et al. [33] are focused on the performance of the selection process rather than the semantic accuracy of the approach. Menzel et al. [34] suggest a step-by-step process to build a concrete, customized evaluation method for any decision scenario. As an application of their framework, they demonstrate an approach for IT infrastructure decisions.

B. *IaaS*

Zhang et al. [6] are limited to IaaS properties such as compute, storage, and network which are eventually ranked by entity costs. Semantically, they claim to perform some basic input validation on the user's query. In [35], the authors present an approach for IaaS cloud selection using MCDM methods. Their method is based on five cost and performance-related criteria for thirteen cloud services. Pawluk et al. [36] introduce an IaaS cloud broker focused on cost minimization. Moreover, they try to address lock-in issues by ranking multi-cloud application scenarios while relaxing the cost objective. Andrikopoulos et al. [37] present a decision support system for assisting the migration between cloud providers again focusing on cost minimization. Gong and Sim [38] suggest a centroid-based search engine with the help of a k-means clustering algorithm for similarity search. In that regard, a user query is transformed into a temporary entity and compared to the existing provider vectors. To mitigate biases caused by missing data, they replace absent data with default, most frequent or similar values. Whereas this approach allows exploring similar providers based on semantic equivalences, it is not feasible for selections targeting portability which warrant must-have requirements. García-Galán et al. [39] present a vendor-specific IaaS selection focusing solely on Amazon EC2 configurations. Based on their feature models, they validate user queries before the optimization process is executed. Jung et al. [40] present CloudAdvisor, a recommender platform focused on QoS properties like cost, performance expectation, and energy efficiency. Similarly, Garg et al. [41] concentrate on a set of quantitative QoS attributes taken from the Service Measurement Index [42]. In addition to academic publications, several web services for IaaS cost comparisons exist[7].

[7]See http://www.planforcloud.com, https://www.cloudorado.com

C. *PaaS*

In Bassiliades et al. [43] the recommendation algorithm of the PaaSport marketplace is discussed. The selection algorithm is a two-step process that first selects all matching vendors based on functional requirements and later ranks them based on nonfunctional requirements using an aggregation scoring function. Whereas the algorithm scores on very specific data values like runtime versions or database storage sizes, it does not use any semantic rules to account for data biases which, as we showed, strongly interrelate with detailed queries. Problems caused by missing data are likely, especially since nonfunctional attributes suggested by the approach such as uptime guarantees or processing cores are not known for a lot of PaaS offerings. Quinton et al. [44] propose a software product lines based approach. They use feature models (FM) to describe different PaaS environments. To bridge the semantic gap between these FMs, a generic cloud knowledge model is introduced to describe all concepts relevant to the domain. Next, several mappings based on the concepts of the cloud knowledge model to features with the same semantics of different FMs are introduced. This approach is comparable to our unified PaaS model complemented by the terminology mappings, but instead all of these rules need to be explicitly specified which requires a lot of manual efforts and risks missing assignments. Surajbali and Juan-Verdejo [45] propose a high-level architecture for a PaaS broker that includes provider selection. Their DSS builds on top of the analytic hierarchy process to recommend and rank available provider alternatives. However, they do not give more insights on how they adapted their formalized selection approach inCLOUDer for PaaS recommendation.

D. *SaaS*

In the field of SaaS, offerings are mainly comparable based on nonfunctional requirements. Godse and Mulik [46] present a SaaS selection focused on Sales Force Automation for CRM, based on the analytic hierarchy process. Schlauderer and Overhage [21] suggest an assessment framework for evaluating software service providers in general. They define and evaluate a set of important assessment criteria and requirements but no selection process by itself.

VI. Limitations and Future Work

Due to the focus on application portability, currently all requirements are equally important for the selection and ordering of the result set. Nevertheless, there exist properties that can be relaxed and weighted with a certain importance dependent on the preferences of a user. We could further enhance our work and add the possibility to specify preferences that serve as user-defined semantic input for the selection algorithms and order the results based on these factors. As of now, all user and application requirements must be specified manually as input for the selection algorithms. In the future, it may be possible to assist the selection process by automating parts of the application requirements detection, e.g., by finding application dependencies through static code analysis.

VII. Conclusion

In this paper, we showed that existing approaches for cloud provider selection lack semantic technologies to account for query and data biases. Both of these factors appear in the process of selecting alternatives from knowledge bases in general and especially in multi-criteria selection scenarios where more dimensions complicate the data and selection scenario. However, current research has not considered these problems appropriately in their works. To that end, we introduced a multi-step selection process including algorithms based on semantic findings applied to the domain of cloud platform selection. We showed that the application of these algorithms improves the accuracy and satisfaction of user queries on the knowledge base by contrasting the results of exact queries with the results of our improved semantic queries on a large data set of real user queries. Whereas we validated our approach for a specific selection domain, the findings and general ideas can be beneficial for a wide range of multi-criteria selection approaches.

References

[1] S. Kolb and G. Wirtz, "Towards Application Portability in Platform as a Service," in *Proc. Symp. Service-Oriented System Engineering*, 2014.

[2] D. Petcu *et al.*, "Portable Cloud applications – From theory to practice," *Future Generation Computer Systems*, vol. 29, no. 6, 2013.

[3] G. C. Silva *et al.*, "A Systematic Review of Cloud Lock-In Solutions," in *Proc. Conf. Cloud Computing Technology and Science*, 2013.

[4] B. Di Martino, "Applications Portability and Services Interoperability among Multiple Clouds," *IEEE Cloud Computing*, vol. 1, no. 1, 2014.

[5] L. Sun *et al.*, "Cloud Service Selection: State-of-the-art and Future Research Directions," *Journal of Network and Computer Applications*, vol. 45, 2014.

[6] M. Zhang *et al.*, "A Declarative Recommender System for Cloud Infrastructure Services Selection," *Economics of Grids, Clouds, Systems, and Services*, vol. 7714, 2012.

[7] J. Lu and D. Ruan, *Multi-objective group decision making: methods, software and applications with fuzzy set techniques*. Imperial College Press, 2007, vol. 6.

[8] E. Triantaphyllou, *Multi-Criteria Decision Making Methods*. Springer, 2000, pp. 5–21.

[9] M. Hogan *et al.*, "NIST Cloud Computing Standards Roadmap," *NIST Special Publication 500-291*, 2011.

[10] Open Grid Forum, "Open Cloud Computing Interface - Infrastructure," 2011.

[11] L. Badger *et al.*, "Cloud Computing Synopsis and Recommendations," *NIST Special Publication 800-146*, 2012.

[12] N. Loutas *et al.*, "Cloud computing interoperability: the state of play," in *Proc. Conf. Cloud Computing Technology and Science*, 2011.

[13] M. Rekik *et al.*, "Anti-Pattern Specification and Correction Recommendations for Semantic Cloud Services," in *Proc. Hawaii Conf. System Sciences*, 2017.

[14] ISO/IEC, "Information technology – Open Virtualization Format (OVF) specification," 2011.

[15] OASIS, "Topology and Orchestration Specification for Cloud Applications Version 1.0," 2013.

[16] C. F. Cargill, "Why Standardization Efforts Fail," *Journal of Electronic Publishing*, vol. 14, no. 1, 2011.

[17] A. Schönberger *et al.*, "Has WS-I's work resulted in WS-* interoperability?" in *Proc. Conf. Web Services*, 2011.

[18] M. Geiger *et al.*, "On the evolution of BPMN 2.0 support and implementation," in *Proc. Symposium Service-Oriented System Engineering*, 2016.

[19] *Software engineering – Product quality – Part 1: Quality models*, ISO/IEC Std. 9126-2, 2003.

[20] *Systems and software engineering – System and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models*, ISO/IEC Std. 25 010, 2011.

[21] S. Schlauderer and S. Overhage, "Selecting Cloud Service Providers – Towards a Framework of Assessment Criteria and Requirements," in *Proc. Wirtschaftsinformatik*, 2015.

[22] L. Kaufman, "Data Security in the World of Cloud Computing," *IEEE Security & Privacy*, vol. 7, no. 4, 2009.

[23] S. Kolb *et al.*, "Application Migration Effort in the Cloud - The Case of Cloud Platforms," in *Proc. Conf. Cloud Computing*, 2015.

[24] ——, "Application Migration Effort in the Cloud," *Services Transactions on Cloud Computing*, vol. 3, no. 4, 2015.

[25] S. Gebus and K. Leiviskä, "Knowledge acquisition for decision support systems on an electronic assembly line," *Expert Systems with Applications*, vol. 36, no. 1, 2009.

[26] F. Pardee *et al.*, *Measurement and evaluation of transportation system effectiveness*. RAND Corporation, 1969.

[27] T. L. Saaty, "How to make a decision: The analytic hierarchy process," *European Journal of Operational Research*, vol. 48, no. 1, 1990.

[28] B. Roy, "Classement et choix en présence de points de vue multiples," *Revue française d'automatique, d'informatique et de recherche opérationnelle. Recherche opérationnelle*, vol. 2, no. 1, 1968.

[29] P. C. Fishburn, "Additive utilities with incomplete product sets: Application to priorities and assignments," *Operations Research*, vol. 15, no. 3, 1967.

[30] S. Kolb and C. Röck, "Unified Cloud Application Management," in *Proc. World Congress on Services*, 2016.

[31] Z. Rehman *et al.*, "Towards Multi-criteria Cloud Service Selection," in *Proc. Conf. Innovative Mobile and Internet Services in Ubiquitous Computing*, 2011.

[32] E. Wittern *et al.*, "Cloud Service Selection Based on Variability Modeling," in *Service-Oriented Computing*. Springer, 2012, vol. 7636.

[33] S. Sundareswaran *et al.*, "A Brokerage-Based Approach for Cloud Service Selection," in *Proc. Conf. Cloud Computing*, 2012.

[34] M. Menzel *et al.*, "$(MC^2)^2$: criteria, requirements and a software prototype for Cloud infrastructure decisions," *Software: Practice and Experience*, vol. 43, no. 11, 2013.

[35] Z. Rehman *et al.*, "Iaas Cloud Selection using MCDM Methods," in *Proc. Conf. e-Business Engineering*, 2012.

[36] P. Pawluk *et al.*, "Introducing STRATOS: A Cloud Broker Service," in *Proc. Conf. Cloud Computing*, 2012.

[37] V. Andrikopoulos *et al.*, "Supporting the Migration of Applications to the Cloud through a Decision Support System," in *Proc. 6th Conf. Cloud Computing*, 2013.

[38] S. Gong and K. M. Sim, "CB-Cloudle: A Centroid-based Cloud Service Search Engine," in *Proc. Conf. Engineers and Computer Scientists*, 2014.

[39] J. García-Galán *et al.*, "Migrating to the Cloud: a Software Product Line based analysis," in *Proc. Conf. Cloud Computing and Services Science*, 2013.

[40] G. Jung *et al.*, "CloudAdvisor: A Recommendation-as-a-Service Platform for Cloud Configuration and Pricing," in *Proc. World Congress on Services*, 2013.

[41] S. K. Garg *et al.*, "A framework for ranking of cloud computing services," *Future Generation Computer Systems*, vol. 29, no. 4, 2013.

[42] J. Siegel and J. Perdue, "Cloud Services Measures for Global Use: The Service Measurement Index (SMI)," in *Proc. SRII Global Conference*, 2012.

[43] N. Bassiliades *et al.*, "A Semantic Recommendation Algorithm for the PaaSport Platform-as-a-Service Marketplace," *Expert Systems with Applications*, 2016.

[44] C. Quinton *et al.*, "Automated Selection and Configuration of Cloud Environments Using Software Product Lines Principles," in *Proc. Conf. Cloud Computing*, 2014.

[45] B. Surajbali and A. Juan-Verdejo, "A Marketplace Broker for Platform-as-a-Service Portability," in *Advances in Service-Oriented and Cloud Computing*. Springer, 2015, vol. 508.

[46] M. Godse and S. Mulik, "An Approach for Selecting Software-as-a-Service (SaaS) Product," in *Proc. Conf. Cloud Computing*, 2009.