Exploring Web Search Engines to Find rchitectural Knowledge

Mohamed Soliman , Marion Wiese†, Yikun Li , Matthias Riebisch†, and Paris vgeriou
Bernoulli Institute for Mathematics, Computer Science and rtificial Intelligence
University of Groningen, Groningen, The Netherlands
{m.a.m.soliman, yikun.li, p.avgeriou}@rug.nl
†Department of Informatics, Universität Hamburg, Germany
Email: {wiese, riebisch}@informatik.uni-hamburg.de

***bstract*—Software engineers need relevant and up-to-date architectural knowledge (K), in order to make well-founded design decisions. However, finding such K is quite challenging. One pragmatic approach is to search for K on the web using traditional search engines (e.g. Google); this is common practice among software engineers. Still, we know very little about what K is retrieved, from where, and how useful it is. In this paper, we conduct an empirical study with 53 software engineers, who used Google to make design decisions using the ttribute-Driven-Design method. Based on how the subjects assessed the nature and relevance of the retrieved results, we determined how effective web search engines are to find relevant architectural information. Moreover, we identified the different sources of K on the web and their associated K concepts.**

***Index Terms*— rchitecture knowledge rchitecture design decisions Search engines**

I. Introduction

rchitectural knowledge (K) is crucial for software engineers to make architectural design decisions [1]. For instance, knowledge about technologies or architectural patterns, including their benefits and drawbacks, is important to select an architectural solution for a design problem. However, finding architectural knowledge (K) is a challenging task [2] for a number of reasons. First, K resides in multiple heterogeneous *K sources*, such as technology documentation [2], issue tracking systems [3], and developer communities (e.g. Stack Overflow) [4]. Thus there is no single source of K that contains all required K.

Second, each source of K contains different *K concepts* (e.g. design decisions, solution alternatives [5], or the benefits and drawbacks of architectural solutions [4]). For instance, developer communities contain predominantly general K concepts [6], such as the benefits and drawbacks of architectural solutions [7]. In contrast, issue tracking systems contain mainly design and reasoning K concepts [6], such as design decisions of existing systems [3]. Thus, depending on the K concept, one may need to look into a different K source.

Third, all K sources are characterized by a fast pace of change, and accelerating technology churn [8]. This makes it even harder for software engineers to find and analyze information within the different sources of K.

One approach to facilitate finding K is to manually capture K (i.e. search for K, and codify it) from multiple sources, and subsequently structure and store it in a repository (e.g. [9]). This supports software engineers to directly find relevant K concepts, without navigating through many sources of K depending on the concept. However, manually capturing K requires significant efforts to gather and keep knowledge up to date; this makes it an expensive means in industrial practice [10].

more pragmatic way to search for K from different sources is to use web search engines (e.g. Google). Web search engines are commonly used by software engineers in their daily business to find technical solutions for problems [11]. Moreover, web search engines can provide access to multiple K concepts, such as design decisions from an existing system (e.g. within open source systems [3]), as well as descriptions of architectural solutions (e.g. within technology documentation [2]). It is even possible to use web search engines to populate K repositories [2].

While web search engines can provide support to find K, recent experiences show that web search engines return many irrelevant results when searching for K [2]. In fact, there is little to no empirical evidence about: a) which K sources and K concepts can actually be found by web search engines; and b) when and how web search engines can be helpful for practitioners.

Our main **goal** in this paper is to *explore which K sources and K concepts are retrieved by web search engines, and to gauge the effectiveness of web search engines to find relevant K during the architectural design process*. To this end, we conducted an empirical study with 53 software engineers with different levels of experience. The subjects used the most popular web search engine (i.e. Google) to find relevant K concepts when conducting the steps of the ttribute-Driven Design (DD) method [12]; DD was chosen as it is one of the most popular architectural design processes in literature. The study results in the following contributions:

We empirically identified K sources, that web search engines are able to find. Moreover, we associated each

This work was supported by ITE 3 and RVO under grant agreement No. 17038 VISDOM (https://visdom-project.github.io/website).

K source with the K concepts they contain, based on the perspective of software engineers. Furthermore, we determined possible correlations between K concepts.

We created a corpus of empirically classified and evaluated web pages and their respective K sources and K concepts.

We measured and compared the effectiveness of web search engines to support software engineers during the execution of the DD steps. Moreover, we determined the most relevant K sources for each DD step according to the evaluation of software engineers.

We identified the K concepts that make web pages highly relevant for software engineers when making design decisions.

The rest of the paper is structured as follows: Section II provides a background on the DD steps, while Section III presents the research questions and study design. Sections IV, V, VI and VII present the results per research question, Section VIII discusses our results and their implications to practitioners and researchers, and Section IX discusses threats to validity. Finally, Section X discusses some related work, and Section XI concludes the paper.

II. TTRIBUTE DRIVEN DESIGN STEPS

Kazman et al. [12] proposed a number of iterative steps within DD to make architectural decisions. For the purposes of our study, we select three of these steps, which are the most *information-intensive* [13]: they require searching for architectural information to be conducted. We explain these three steps and the type of information that software engineers need to perform them.

Identify design concepts: In this step, alternative architectural solutions are identified for a design issue. For example, a software engineer might look for alternative broker technologies, which might fulfill system requirements and align with the system constraints. s an example, these alternative solutions for broker technologies could be RabbitMQ, Kafka and ctiveMQ. To perform this step, software engineers need to search for information regarding alternative solution options.

Select design concepts: In this step, one architectural solution is selected from a list of alternative solutions (from the previous step). This is done by comparing alternative solutions with each other regarding their ability to fulfill functional requirements and quality attributes. For example, RabbitMQ, Kafka and ctiveMQ are compared regarding performance and reliability to decide on the most suitable broker technology. To perform this step, software engineers need to search for benefits and drawbacks of the alternative options, e.g. information about performance benchmarks and reliability features for each of the alternative solutions.

Instantiate architecture elements: In this step, the selected architectural solution (from the previous step) is customized to match the system requirements. For instance, to achieve high availability, replication tactics need to be implemented. This is done by configuring the selected broker technology (e.g. Kafka) to add a specific number of replicated instances. To perform this step, software engineers need to search for information and experiences regarding technology features, and their abilities to implement architectural tactics [12].

III. STUDY DESIGN

. *Research questions*

To achieve our goal (see Section I), we ask the following research questions (RQs):

(RQ1) *Which K can web search engines support to find?*

> ***(RQ1.1)*** *Which K sources can web search engines find?*
> ***(RQ1.2)*** *Which K concepts are prominent within the found K sources?*

s discussed in Section I, researchers have explored multiple different K sources and their K concepts. However, there is no comprehensive list of K sources and K concepts on the web. Thus, we ask RQ1.1 to determine possible sources of K that exist on the web. nswering RQ1.1 can confirm the K sources that are already known but it can also reveal new K sources, which have not been previously explored. Moreover, we ask RQ1.2 to determine the K concepts, which commonly appear in each of the found K sources.

(RQ2) *Which K concepts co-occur on the web?*

We ask RQ2 to determine possible relationships between K concepts on the web. nswering RQ2 can help us determine if K concepts appear together on the web similarly to their conceptual relationships in K ontologies (e.g. architectural solutions have benefits and drawbacks [4]). This can support assessing whether the relations between concepts in existing K ontologies are reflected on the K found in the web.

(RQ3) *How well do web search engines support software engineers in following the DD steps?*

> ***(RQ3.1)*** *How effective are web search engines to find K needed for performing the DD steps?*
> ***(RQ3.2)*** *Which K sources have the biggest contribution on the effectiveness of web search engines for each of the DD steps?*

Within each step of the DD (see Section II), software engineers need to search for different types of architectural information. We ask RQ3.1 to quantitatively measure the effectiveness of web search engines to find relevant architectural information, and to determine how much web search engines can support software engineers during the different architectural design steps. Moreover, we ask RQ3.2 to determine the most useful sources of K for each DD step. nswering RQ3.2 can support prioritizing and directing our future research efforts to explore and capture K, by focusing on certain K sources, that yield the highest benefit to software engineers.

(RQ4) *Which K concepts make web pages more relevant for design decisions?*

When using web search engines to perform architectural design tasks, some K concepts may increase the relevance of the corresponding web pages. We ask RQ4 to determine which K concepts indeed can make a web page more relevant for

T BLE I: Industrial experience of participants

Software development		Software architecture	
# Years	# Participants	# Years	# Participants
>10 Years	4	>5 Years	5
3-10 Years	13	2-5 Years	9
3 Years	36	1 Year	39

specific architectural tasks than others. nswering RQ4 could provide guidance on which K is worth sharing on the web.

B. Overview on the research process

To answer the RQs, we conducted an exploratory case study [14] with 53 software engineers (see Section III-C) who used the most popular web search engine (i.e. Google) to perform the DD steps (see Section II). The conducted case study is an embedded case study, where the DD steps constitute our case and the executed search queries from the 53 software engineers are the units of analysis.

To collect data, we asked the 53 software engineers to solve six architectural design searching tasks (see Section III-D) using Google, where each searching task performs one of the three DD steps [12], as explained in Section II. For each searching task, the participants executed multiple queries in Google, and assessed the resulted web pages regarding two aspects: 1) The relevance of each web page to the searching task, and 2) The K concepts which exist in each web page. Further details are presented in Section III-E.

To analyze the collected data, and answer the RQs, we used the following analysis methods:

Web pages classification: To answer RQ1.1, RQ1.2 and RQ3.2, we classified collected web pages (retrieved by Google for each query) into their respective K source using a semi-automated approach (see Section III-F).

Descriptive statistics and correlations: To answer RQ1.2, RQ2 and RQ 3, we used descriptive statistics and evaluated correlations between K sources, K concepts and relevance of web pages using Pearson $\tilde{\chi}^2$ test [15] (see Section III-H).

Effectiveness measurement: To answer RQ3.1 and RQ3.2, we measured the effectiveness of Google using standard information retrieval metrics (see Section III-G).

C. Participants of the case study

The participants of the case study are 53 software engineers. 50 of the participants attended a software architecture master course at the University of Hamburg, and 3 additional software engineers volunteered to participate in the study. n overview on the industrial experience of the participants is presented in Table I. dditional information regarding the technical background of the participants is available online.

D. Searching tasks

To answer the RQs, each participant (see Section III-C) solved three searching tasks, where each task corresponds to one of the three DD steps (see Section II).

To support the validity of our results, we have designed two searching tasks for each DD step. Thus, we have in total six tasks, from which we have randomly assigned three tasks (one task per DD step) to each participant. The six searching tasks are real design problems, which have been gathered based on interviews with practitioners in a previous study [16] within the field of architectural knowledge. Table II presents a brief description of the six searching tasks, and their relationship to the DD steps. complete description of each task is available online.

T BLE II: rchitectural searching tasks

DD step	ID	Task description
Identify design concepts	T1	For a realtime dashboard, identify middleware technologies which scale to >100k users
	T2	system needs to communicate with mobile apps. Identify JSON parsers for Java with high performance, considering license constraints.
Select design concepts	T3	system communicates with a knowledge base via publish/subscribe patterns. Compare interoperability and latency of RabbitMQ, Kafka, and ctiveMQ.
	T4	Compare three technology families for big data systems: data collector, message brokers, and ETL engines. Requirements are throughput of 15,000 events/sec and availability of 99.99%.
Instantiate architecture elements	T5	CRM apps communicate with other systems using pache Camel and RabbitMQ. Search for technology features and components designs to determine mechanisms channeling, translation and routing, and deployment topology.
	T6	n application exposes services to other apps. Search for best practices regarding service decomposition to achieve high cohesion and low coupling.

E. Case Study procedures

1) Preparations before the study: To support the validity of our results, it is important to ensure that the participants have a clear understanding about the procedures of the study, as well as the searching tasks, and K concepts. Therefore, the authors met with the participants in two sessions and explained the study procedures, as well as the assessment of web pages regarding their relevance and K concepts. fter the first session, each participant received a user-guide (available online) with details about the study and a video tutorial on how to perform the tasks during the study. t the beginning of the second session and before conducting the study, the authors made a demo on assessing the relevance and specifying K concepts for an example web page. In this demo, the participants were asked to specify the relevance and K concepts for this example web page and provide their input via a polling feature in a web-conferencing tool. The polling results were discussed with the participants to show them how to correctly assess the relevance and how to specify K concepts for each web page e.g. by paying particular attention to the requirements and constraints of the tasks.

We provide below the definitions for the degrees of relevance, that participants could choose from (in a five-level Likert scale):

Very High Relevance (VH): The web page discusses a similar problem to that of the task and contains useful information. The web page provides an answer to the

searching goal, and helps with fulfilling more than one requirement of the task.

High Relevance (H): The web page addresses a similar problem to that of the task and contains useful information. The web page provides an answer to the searching goal, and helps with fulfilling one requirement of the task.

Medium Relevance (M): The web page addresses a different problem to that of to the task at hand, but it provides some relevant information to the task, which could be an answer to the searching goal. Nevertheless, the provided information does not match specifically the task's requirements.

Low Relevance (L): The web page contains information, which is only remotely relevant to solving the given task, but might help for refining the search.

No Relevance (N): The web page has nothing to do with the task. It has no relevant information.

Moreover, each participant specified certain K concepts for each web page. The list of K concepts has been derived from existing literature [4], [17], [18] and is as follows:

Solution description: general information on an architectural solution.

Solution alternatives: multiple (alternative) architectural options for a certain design issue. The architectural options could be listed in the text or as a comparison of different options.

Solutions benefits: information about the advantages of certain architectural solutions.

Solutions drawbacks: information about the disadvantages of certain architectural solutions, even discouraging their application.

Made design decisions: explanation about the architecture of a specific system. This includes the description of an existing architectural design of a specific system, or the explanation about certain design decisions of a specific system.

Others: other relevant architectural information.

2) Study execution: We asked the participants to perform the tasks in the order given to them. To ensure that the sequence of tasks does not influence the study, we provided each participant with a different sequence of tasks (available online). Moreover, we asked each participant to perform at least three queries per task and to evaluate the top 10 Google results for each query, as most users do not assess more than the top 10 results on the first page.

To facilitate specifying the relevance and K concepts for each web page, we developed a Google Chrome plugin (provided online), which offers a user interface and stores submitted relevance and K concepts in a database.

The participants started the study in a synchronous web-conference meeting, where they were able to directly ask for clarification if they had any uncertainties. By the end of the web-conference, the participants continued solving the tasks independently, while using the provided plugin to specify the relevance and K concepts for each web page.

3) fter the study: fter solving all the tasks, the participants were asked to fill out an exit survey (available online). We asked the participants about their experience searching for architectural information using Google, as well as the complexity to analyze web pages regarding K concepts.

F. Web pages classification

s a result of the study, we received 5175 web pages (with 2623 unique pages) from executing 477 queries in Google, where each web page is evaluated from a participant regarding its relevance and K concepts. To answer RQ1.1 and support answering the other RQs, we analyzed the resulted unique web pages to determine the K source (e.g. forum, blog) for each page. To achieve this, we followed a semi-automated approach using two main steps, which are explained in the following sub-sections.

1) utomatically clustering URLs of web pages: We executed a clustering algorithm [19] on the list of URLs to determine initial clusters of web pages. We have decided on this algorithm due to its excellence in clustering short text, compared to other classical clustering and topic modeling algorithms (e.g. LD). Before executing the clustering, we filtered the URLs regarding symbols and stop words, and differentiated between host name and path within a URL.

We started the clustering algorithm using a big number of clusters (i.e. 100 clusters), and then reduced the number of clusters gradually after checking the results to reach the best possible clusters of URLs. Starting with a big number of clusters facilitated determining the commonalities between smaller groups of URLs, which could be later aggregated into a single cluster. fter several iterations of clustering, we achieved the best results by having 31 clusters.

By executing the clustering algorithm on the 2623 unique URLs, it succeeded to split the URLs into 27 consistent clusters with 1280 URLs (e.g. one consistent cluster is all blog pages in pache websites), and 4 inconsistent clusters (i.e. mixed of different K sources) with 1343 URLs. This separation has been determined by inspecting samples of web pages of the URLs within each cluster.

2) Manually classifying web pages: To determine the K source for each web page, we started by analyzing the 27 consistent clusters to determine dominant K sources in each cluster. Manually classifying web pages in the consistent clusters was done by inspecting sample web pages from the cluster to determine the dominant K source category of this cluster. s a result of this step, we identified 15 initial categories of K sources, and classified the 1280 URLs within the consistent clusters.

Based on the 15 initial categories of K sources, the first three authors manually categorized the rest of the URLs (1343 web pages from within the 4 inconsistent clusters) into their respective categories of K sources. This has been done by inspecting each of the web pages. While manually inspecting the web pages, we ignored offline web pages, spam and web pages in languages other than English or German. Moreover, a cross-check validation has been conducted between the first

and second authors, as well as between the first and third authors to ensure agreement on the classification. To ensure agreement between the authors, we merged the 15 initial categories into 9 categories.

s a result of this step, we identified categories of K sources on the web (see Section IV-). Moreover, we created a corpus of 2522 unique web pages, which are categorized based on their K sources.

G. Measurement of effectiveness

To answer RQ3.1 and RQ3.2, we measure the effectiveness of Google using two metrics: $Precision@k$ and *Normalized Discount Cumulative Gain*($nDCG@k$), where k is the maximum number of search results that are considered for evaluation. We considered k from 1 to 10 in our evaluations.

$Precision@k$ [20] is the ratio between the number of relevant web pages (low or medium or high or very high), by the number of retrieved web pages in the results.

The ranking and relevance of the retrieved web pages are important factors to assess the effectiveness of search engines. However, Precision does not consider the ranking and relevance of web pages. Therefore, we use $nDCG@k$ [20], which consider both the ranking and relevance of the retrieved web pages. $nDCG@k$ is a well known metric in information retrieval and has been used successfully in software engineering research (e.g. [21]).

The main idea of $nDCG@k$ [20] is to compare the ideal ranking of web pages ($IDCG$) to the ranking retrieved from a search engine for a certain query (DCG). For example, consider a task for which two users execute a query: the first user rates the top three web pages with relevance 3, 3, 1 while the second user rates the top three web pages with relevance 2, 2, 1. Meanwhile, the ideal ranking for this task is 3, 3, 2 when evaluating the top three search results. The $nDCG@3$ will compare the rankings 3, 3, 1 and 2, 2, 1 against 3, 3, 2.

To compare the rankings of queries with the ideal ranking, we divide the $DCG@k$ of each query with the $IDCG@k$ (the $DCG@k$ of the ideal ranking). The ideal ranking is based on combining the individual rankings of web pages (based on their relevance to a task) from all participants in the study.

In order to translate the importance of relevance and ranking into a metric, we calculate the $DCG@k$ for each ranking (from each query). The $DCG@k$ provides different weights for web pages based on their relevance and ranking in the list of results (i.e. the higher the relevance and rank, the more weight). One common way to implement the $DCG@k$ is to use the logarithmic scale to provide the right weight based on the relevance and ranking. Specifically, we calculate $DCG@k = \sum_{i=1}^{k} \frac{2^{r\ l} \quad 1}{log_2\ i+1)}$, where rel_i is the degree of relevance of a web page found by a query (based on the Likert scale defined in Section III-E1).

To calculate the $nDCG@3$ for the previous examples q1 and q2, we calculate the $DCG@k$ for the rankings 3, 3, 1 (ranking of q1), 2, 2, 1 (ranking of q2) and 3, 3, 2 (the ideal ranking).

H. Statistical analysis and significance tests

To answer RQ1.2, RQ2 and RQ4 we used descriptive statistics. Moreover, we executed Pearson $\tilde{\chi}^2$ correlation tests [15] to determine the most relevant relationships.

To compute Pearson $\tilde{\chi}^2$ correlation tests and measure the phi coefficient we used the statistics tool SPSS[1]. Due to the large sample size, all correlations are significant (even to the 1% level), but only the phi coefficients measured for RQ2 revealed correlations with effect sizes greater than 0.3 which means medium strong or strong correlations [22].

To answer RQ1.2, we used descriptive statistics to determine relationships between K sources (based on our classification see Sections III-F and IV-), and the K concepts (as specified by practitioners, see Section III-E1). For this, we counted the occurrences of each K concept on the web pages and aggregated this for each K source separately.

To answer RQ2, we performed a two-tailed T-test [23] with $= 0.05$ on all values of $Precision@k$ and $nDCG@k$ for each of the three DD steps separately. Executing the T-test requires determining if the calculated values of $Precision@k$ and $nDCG@k$ have equal or unequal variances. Thus, we executed an F-test [23] with $= 0.05$ to determine if the values of $Precision@k$ and $nDCG@k$ have equal or unequal variances. Based on the results of F-test, we have executed the right method of T-test, either with equal or unequal variances.

We answered RQ4 using descriptive statistics to present the distribution of relevance within each K concept, i.e. we evaluated only the web pages that contain the respective K concept. Both the K concepts and relevance of web pages have been specified by the participants during the study (see Section III-E1).

IV. RQ1: RCHITECTURE KNOWLEDGE SOURCES ND CONCEPTS ON THE WEB

. RQ1.1: K sources on the web

Table III shows the identified K sources, which have been retrieved by Google when searching for architectural information. Moreover, Table III shows the percentages of each category based on our corpus (2522 web pages). **The results suggest that the predominant categories are blogs and tutorials as well as technology vendor documentations** (and to a lesser extent scientific contents). During our manual classification (see Section III-F), we found a wide variety of blogs, such as personal blogs (e.g. Martin Fowler's blog[2]) and technology blogs (e.g. S P technology blog[3]). Moreover, technology vendor documentations come at different levels of detail, from a high level description of technology to detailed code specification (e.g pache technologies[4][5]).

On the other hand, **source code repositories (e.g. Github) and knowledge repositories are not commonly found** in the

[1] https://www.ibm.com/de-de/analytics/spss-statistics-software
[2] martinfowler.com
[3] blog.sap-press.com
[4] kafka.apache.org/documentation
[5] flink.apache.org

T BLE III: K sources on the web

K source	Example	%
Blogs and tutorials	dzone.com	39%
Technology vendor documentations	metamug.com/docs	23%
Scientific contents	ieeexplore.ieee.org	13%
Forums	stackoverflow.com	7%
Technical books and white papers	livebook.manning.com	4.5%
Source code repositories	github.com	4.5%
Knowledge repositories	stackshare.io	4%
Presentations and videos	slideshare.net	2.7%
Others (e.g. tools, patents)	google.com/patents json.parser.online.fr	2.3%

T BLE IV: Correlation between each of the K concepts

	Descr.	ltern.	Benef.	Drawb.	Decisions
Description	1.000	0.181	**0.347**	0.203	0.222
lternatives		1.000	**0.355**	**0.340**	0.079
Benefits			1,000	**0.651**	0.157
Drawbacks				1.000	0.122
Decisions					1.000

T BLE V: Co-occurrences between Benefits and Drawbacks

	Drawbacks		
Benefits	does not contain	contains	
does not contain	3203	46	3249
contains	820	1106	1929
Sum	4023	1152	5175

search results, and they come in less variety. Furthermore, the results show that **issue tracking systems (e.g. pache issue trackers[6]) are not retrieved at all by Google** when searching for architectural information.

B. RQ1.2: Prominent K concepts in the K sources

Figure 1 shows the percentages of occurrence for each of the K concepts in the found K sources (see Section IV-). Because each web page can contain multiple K concepts, the sum of percentages for all K concepts in an K source can exceed 100% (for more details see the online resources. Based on Fig. 1, we can observe the following:

Solution alternatives, benefits and drawbacks are less present within the technology vendor documentations and source code repositories; presumably this is because these K sources usually discuss a single architectural solution, such as the documentation or source code of a specific technology.

Made design decisions are underrepresented in all K sources. Thus, finding concrete examples of design decisions (e.g. a decision on specific components of a system), and their rationale for existing systems are not as easy to find among the K sources on the web.

Solution descriptions are more prominent than the other K concepts within technology vendor pages. This is probably because this K source focuses on describing a certain architectural solution in more detail.

V. RQ2: Co-occurrences of K concepts

In order to evaluate the co-occurrences of K concepts in web pages, we measured the correlation between the K concepts (as explained in Section III-H). Table IV shows the correlation coefficients between each pair of K concepts.

From Table IV, we can observe that **the correlation between benefits and drawback stands out with a correlation coefficient of 0.651** which means a strong correlation exists [22]. This means that either a) web pages containing benefits often also contain drawbacks or b) web pages containing drawbacks often also contain benefits. To determine the right interpretation for the correlation between benefits and drawbacks, we inspected the exact frequencies of co-occurrences between benefits and drawbacks (see Table V).

[6]issues.apache.org

We can observe that drawbacks are rarely presented without benefits (46 out of 1152), while there are more web pages containing benefits without drawbacks (820 out of 1929). This means that **it is common to find web pages with benefits and no drawbacks, while it is rare to find web pages with drawbacks and no benefits**. This can be also seen in Fig. 1. For example, blogs and tutorials contain nearly twice the amount of benefits compared to drawbacks. This indicates that software engineers in communities tend to praise the benefits of technologies rather than realistically evaluate the pros and cons of technologies equally.

From Table IV, we can also observe that the correlations between alternatives and benefits, as well as between alternatives and drawbacks (0.355 and 0.340 respectively) are also medium strong [22]. This indicates that **lists of alternative solutions, are usually accompanied with a comparison between them regarding their benefits and drawbacks**. One common example for this are knowledge repositories and forum entries containing alternatives, with their drawbacks and benefits.

final observation from Table IV, is a medium strong correlation (0.347) [22] between solution descriptions and benefits. This is quite common in technology vendor pages, where they commonly describe their technologies and their benefits, while omitting their drawbacks (see Fig. 1).

VI. RQ3: Se rch engines support for DD steps

. *RQ3.1: Effectiveness of search engines to support DD*

Figure 2 shows the average $nDCG@k$ and $Precision@k$ when searching for architectural information during the three DD steps. From Fig. 2, we can observe that the maximum average $Precision@1$ is 0.75 (for the "Select design concepts" step), which means that on average 75% of the queries retrieved a relevant (low, medium, high, very high) web page at the top result (k=1) from Google. On the other hand, the lowest $Precision@10$ is 0.48 (for the "Identify design concepts" step), which means that on average 48% of the retrieved top 10 web pages were relevant (i.e. have relevance of low, medium, high, very high); in this case, web search engines return both relevant and irrelevant results equally.

To compare the differences between DD steps regarding their $nDCG@k$ and $precision@k$, we executed a significant

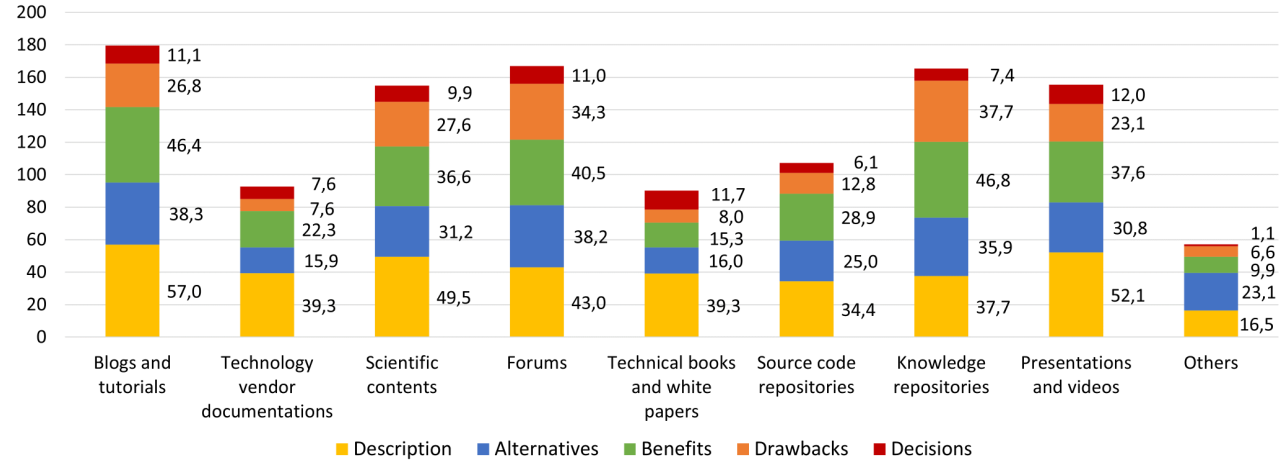

Fig. 1: Distribution of K Concepts per K source

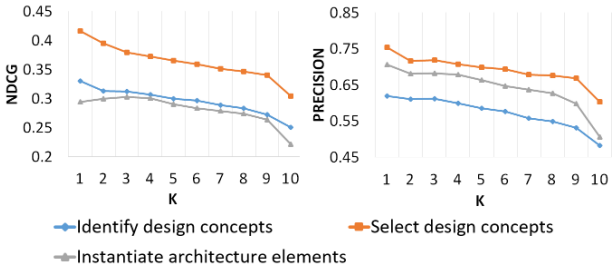

Fig. 2: verage nDCG and precision in finding architectural information for each DD step.

T-test (as explained in Section III-H). The detailed results of the tests are available online.

From the test, **we found that the "Select design concepts" step has significantly higher** $nDCG@k$ **for k between 1 and 9 (**$nDCG@k_{1\rightarrow 9}$**) compared to the other two design steps**, while there is no significance difference in $nDCG@k_{1\rightarrow 1}$ between the "Identify design concepts" and "Instantiate architecture elements" steps. For example, from Fig. 2 the average $nDCG@1$ for the "Select design concept" DD step is 0.42, while the average $nDCG@1$ is 0.29 for the "Instantiate architecture elements" DD step. This means that on average 42% of the "Select design concept" queries retrieved highly relevant (i.e. very high) web pages at the top result from Google, compared to just 29% for the "Instantiate architecture elements". Thus, **finding K to select an architectural solution from alternatives is easier than finding K to instantiate a certain architectural solution**.

The significance test also showed that the "Select design concepts" step has significantly better $precision1 \rightarrow 9$ compared to the "Identify design concepts". In contrast, there is only a slight difference of $precision1 \rightarrow 10$ between the "Select design concepts" and "Instantiate architecture elements".

Looking at both the $nDCG@k$ and $precision@k$, on the one hand we can notice that the "Identify design concepts" has significantly lower $nDCG@k$ and $precision@k$ compared to the "Select design concepts". This means **it is harder to find K on alternative architectural solutions than to find information on how to compare them**. On the other hand, the "Instantiate architecture elements" step has significantly lower $nDCG$ compared to the "Select design concept" step but a comparable $Precision$. Because $nDCG$ considers the ranking and relevance of results compared to $Precision$, the $nDCG$ and $Precision$ of the "Instantiate architecture elements" indicate that **Google finds many distantly relevant web pages (e.g. general concepts on components design) to support the "Instantiate architecture elements" step**. In contrast, it is challenging for Google to find highly relevant solutions, which fulfill specific requirements.

B. RQ3.2: The most influential K sources

Figure 3 shows the top six K sources (see Table III) with the highest effectiveness in each of the DD steps. We can observe the following:

Blogs and tutorials have the biggest contribution on the effectiveness of Google in all three DD steps. Thus, **blogs have the highest relevance and highest ranking, and show to contain the most useful K compared to other sources**.

***Technology vendor documentation* shows to have the second highest contribution on the effectiveness of Google for both the "Identify design concepts" and the "Instantiate architecture elements" steps**. This indicates their usefulness to identify options for architectural solutions. Moreover, technology vendor documentation contains detailed information regarding the technology features, which are useful for the "Instantiate architectural elements" step. s shown in Section IV-B solution alternatives and solution drawbacks are underrepresented in technology vendor documentation, which may be the reason why they are not as effective for the "Select design concepts" step.

***Knowledge repositories* (e.g. [9]) show to be effective for** $@k_{1\rightarrow 2}$ **(i.e. for the top two Google results) only within the "Select design concepts" step**, while it has a negligible contribution on the effectiveness within the other two DD steps. This is because most knowledge repositories on the web (e.g. stackshare) contain mainly high level information

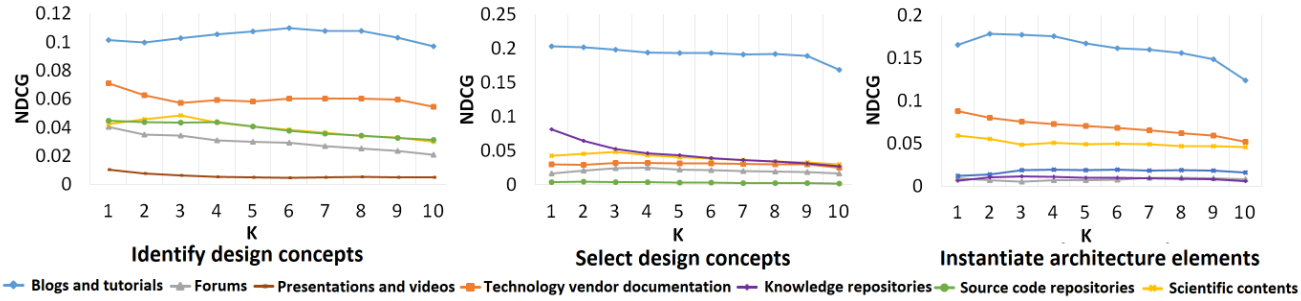

Fig. 3: The contribution of each K source in the effectiveness of Google per DD step

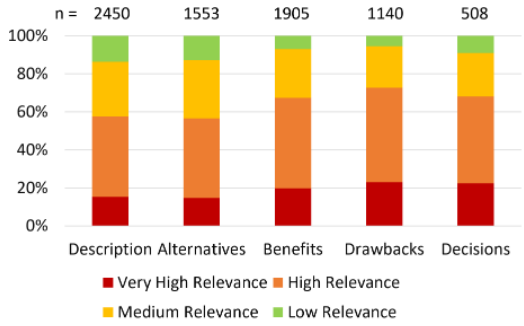

Fig. 4: Distribution relevance per concept

to compare multiple technologies with each other; they can thus somewhat help with "Selecting design concepts".

Scientific contents **(e.g. papers or thesis) show to have a higher contribution on the effectiveness of Google than forums and source code repositories, for the "Instantiate architecture elements" step**. This could be due to the scarcity of K regarding component designs in forums and source code repositories, while scientific contents are rich with component designs (e.g. reference architectures).

Forums and source code repositories **show to have a limited influence on the effectiveness of Google of searching within the three DD steps**. This might be because both forums and source code repositories are not well found by Google (see Table III) compared to the other K sources.

VII. RQ4: K CONCEPTS IN HIGHLY RELEV NT WEB P GES

Figure 4 shows the total number of relevant web pages (presented as "n" in the figure), and the percentages of the different degrees of relevance (low, medium, high and very high as rated by the participants), in which K concepts appear. From Figure 4, we can observe that **web pages that contain solution benefits, solution drawbacks and made design decisions are most relevant for finding K**, while web pages with solution description' and solution alternatives have a lower probability to be high or very high in relevance. For example, a web page containing solution drawbacks has a higher probability to be rated high or very high in relevance (72.7%) than a web page containing solution alternatives (56.5%). (Details are available online).

Even though, solutions drawbacks and made design decisions are not found by Google in its results as often as solution descriptions and solution alternatives, they showed to be the most common K concepts in highly relevant pages, and thus they could be the most important for making design decisions.

VIII. DISCUSSION

. *RQ1: K sources and concepts on the web*

1) Implications for practitioners: The list of K sources on the web (see Table III) and their respective K concepts (see Fig. 1) could guide practitioners to determine the scope of searching (i.e. to search in the whole web or in specific web sites). For example, forums like Stackoverflow have shown to contain useful K (e.g. [7], [24]) about the benefits and drawbacks between architectural solutions (see Fig. 1). However, forums are not well considered by Google (see Table III). Thus, practitioners should focus the scope of searching on specific forums like Stackoverflow to find more K on the benefits and drawbacks between solutions.

2) Implications for researchers: The list of K sources in Table III provides an overview of possible web sources of K, from which researchers could extend current approaches to document K (e.g. [2]). On the one hand, blogs are well indexed by Google but not previously explored by researchers for K. Thus current studies on K (e.g. [7], [24]) could be extended to explore the K in blogs. On the other hand, some K sources are previously explored by researchers for K (e.g. forums [24] and issue tracking systems [3]), but they are not well indexed by Google. These K sources deserve extra attention to improve their ranking on web search results.

The distribution of K concepts in each K source in Fig. 1 show that there is no single K source which contains specific K concepts. However, certain K concepts come more often in certain K sources. Thus, researchers could make use of this information to develop K documentation approaches, which consider multiple K sources. For example, an K approach can find and document benefits and drawbacks from forums, and solution description from technology vendor documentations.

B. *RQ2: Co-occurrences of K concepts*

1) Implications for practitioners: The results in Section V show that benefits are preconditions for the occurrence of drawbacks in web pages, and that drawbacks rarely come alone

in a separate web page. The results increase the awareness of practitioners that the predominant presence of benefits on the web does not mean lack of drawbacks for architectural solutions; it rather means the rarity of drawbacks on the web. To resolve this problem, practitioners should use web search engines to search explicitly for the drawbacks of architectural solutions.

2) Implications for researchers: The co-occurrences between K concepts in Section V shows that K concepts do not come all together in a single web page. However, subsets of K concepts co-occur together more often than others. For instance the correlations between K concepts in Table IV could be grouped into three subsets of K concepts: (benefits and drawbacks), (alternatives, benefits and drawbacks), and (Solution description and benefits). Researchers need to consider this division of K concepts on the web, when developing approaches to automatically capture K (i.e. search for K, and codify it), e.g. developing dedicated K capturing approaches for each subset of K concepts.

C. RQ3: How search engines support the DD steps

1) Implications for practitioners: The results in Section VI- verified previous experiences with practitioners [2] that web-search engines return many irrelevant results. However, our study provides the first precise evidence on the effectiveness of Google to find K for each DD step. The differences in the effectiveness between the DD steps (as shown in Fig. 2) can help increase the awareness regarding the expected relevance of the retrieved results. This can guide practitioners to determine when to rely on web search engines, and when to seek other ways (e.g. asking experts) to search for K. For instance, practitioners could rely on search engines to find K for the "Select design concepts" step. But, they are better off seeking other ways during the "Instantiate architecture elements" step, as many of the retrieved results from web search engines are distantly relevant.

2) Implications for researchers: The results in Section VI- support prioritizing requirements for K management approaches, and especially about which DD steps to support better. For instance, researchers should give higher priority to extending K capturing approaches for the "Identify design concepts" and the "Instantiate architecture elements" steps, as Google has lower effectiveness in supporting these steps. This is probably because Google cannot determine the context of software engineers when identifying or instantiating architectural solutions. To support the "Identify design concepts" step, researchers could propose approaches that relate business requirements, design issues and alternative architectural solutions. Moreover, to support the "Instantiate architecture elements" step, design decisions and their rationale should be captured from existing systems and shared with practitioners.

The results in Section VI-B verify that blogs and tutorials are valid options for exploring and capturing K, because they are the most relevant and highly ranked. However, during our web pages classification (see Section III-F), we found that blogs and tutorials are quite diverse (e.g. private versus company blogs). This makes it very challenging to apply information retrieval or extraction techniques on blogs. Thus, we propose first exploring the different types of blogs and tutorials and the K concepts inside them. Moreover, the results in Section VI-B can support developing specialized software architecture searching approaches. One idea is to re-rank results of Google differently for each DD step; this can be achieved by developing heuristics based on the effectiveness of K sources to support DD steps (see Fig. 3).

D. RQ4: K concepts in highly relevant web pages

1) Implications for practitioners: Section VII shows that solution drawbacks and made design decisions are the most frequently appearing K concepts within highly relevant web pages. This indicates the importance of solution drawbacks and made design decisions for practitioners. On the one hand, solution drawbacks provide K on when an architectural solution could be discouraged. This is important for practitioners to prevent selecting architectural solutions, which must be replaced later due to their drawbacks. Replacing architectural solutions after their implementation often requires substantial effort. On the other hand, made design decisions can be potentially reused by practitioners. Thus, practitioners should share more K about solution drawbacks and their made design decisions on the web (e.g. in blogs or white papers); this can be done in academic or industry conferences, or by creating dedicated community websites for this type of K.

2) Implications for researchers: The results in Section VII support prioritizing K management approaches to focus on important K concepts with the highest relevance to practitioners. For example, researchers should consider automatically capturing K related to made design decisions and solution drawbacks more than other K concepts, because they contain the most relevant K for making design decisions. Developing such approaches can be challenging because both made design decisions and solutions drawbacks present minorities compared to other K concepts on the web (see Fig. 1). Thus, using classification algorithms could be useful to filter web pages on the web with drawbacks or decisions.

IX. THRE TS TO V LIDITY

1) Construct validity: In our study in Section III-E2, the experience and background of participants (see Table I) might have influenced their assessment of web pages (i.e. specifying the relevance and the K concepts). To mitigate this, the participants received training, and materials (as a user guide and video). In addition, the participants were accompanied by the researchers at the beginning of the study. lso, since participants solved some tasks on their own time, we did not have full control of their behavior (e.g. fatigue). However, we tried to mitigate this by changing the sequence of tasks for each participant.

nother threat of validity is the possibility of mistakes in gathering data from participants during the study. To mitigate this, the participants used a plugin, which captured their input and stored it in a database for analysis. In this way, we were

able to verify the data for any possible mistakes, and validate it with the participants.

2) Reliability: The classification of web pages into their respective K sources presents a threat to reliability. However, we tried to ensure consistency in the classification, by establishing categories that have the highest agreement between the first three authors of the paper. Moreover, to facilitate replicating the analysis, we provide our corpus online.

3) External validity: Our study used Google as our case for a search engine, without exploring other search engines (e.g. Bing or Baidu). However, Google is the most popular search engine, and thus our results could be generalized on most users. Our limited number of tasks (six tasks in Table II) compared to the high variety of architectural tasks in practice is another threat to the external validity. To partially mitigate this, we have designed two tasks for each DD step to reduce the dependency on a single task. Finally, the limited number of participants (see Table I) in the study is another threat to the external validity of results. However, the participants have different backgrounds and experiences, which supports, to some extent the generalizability of results.

X. Rel ted Work

We are not aware of any previous studies on web searching approaches to find K. Thus, our study is the first to investigate using web searching to perform architectural tasks. In this section, we discuss some related work in the fields of K, as well as studies on web searching in software engineering.

rchitectural knowledge. Researchers in the field of K have explored the main K concepts, such as design decisions [18], their types [25], rationale of decisions [26] and solutions alternatives [5]. These studies established the fundamental K concepts, which we investigate in our presented study. However, they do not propose approaches for capturing or finding K.

Some approaches propose catalogs and repositories of K to facilitate structuring and sharing K. For example, Elmalki and Zdun [27] modeled common types of DDs for microservice architectures. Malakuti et al. [28] created a catalog for the types of DDs when designing IOT system. While they can guide software engineers during design space exploration, they require extra manual effort to find and codify K.

Recent efforts on K propose approaches to automatically capture K from different K sources using machine learning and information retrieval techniques. For example Gorton et al. [2] proposed an approach to identify K in technology documentation, and specially identify documents with certain architectural tactics (as one architectural solution). Bhat et al. [3] captured K from issue tracking systems. They especially captured and classified the different types of design decisions (as one K concept) in issue tracking systems. Soliman et al. [16] improved the effectiveness of searching for K in Stackoverflow. The approach uses machine learning and heuristics to re-rank the results of search engines. However, all three approaches do not study the effectiveness of web searching to support architectural activities.

Web searching in software engineering. Some approaches empirically investigated the usage of web searching by software engineers. Xia et al. [11] empirically determined the most common and most complex software engineering tasks (e.g. searching for solutions to bugs or for third party libraries) using web search engines. Rahman et al. [29] studied the effectiveness of web searching when searching for source code on the web. Their results show that searching for source code is harder than searching for other types. Hassan et al. [30] empirically analyzed web search queries and results related to code exceptions. Moreover, they proposed an approach to capture knowledge related to code exceptions from the web. However, these three studies do not consider finding K to perform software architectural tasks.

Other approaches tried to improve the effectiveness of retrieving code examples from the web. For example, Wang et al. [31] proposed an approach to capture code examples regarding PI usage from the web. Sirres et al. [32] proposed an approach to improve the effectiveness of source code retrieval by augmenting queries with knowledge from Stackoverflow and Github. Their results improve the effectiveness of source code searching compared to Google. However, their approaches focus on finding source code rather than K.

XI. Conclusion nd Future Work

Our main goal in this paper is to explore the retrieved architectural knowledge (K) from the web, and the effectiveness of web search engines, when performing three ttribute-Driven Design (DD) architectural steps. To achieve our goal, we conducted an exploratory study with software engineers, who used Google to find K for making design decisions. Our results provide several interesting results. First, we provide an overall view on the different sources of K and their associated K concepts on the web. This can be useful to extend current K capturing approaches. Second, we determined the differences in the effectiveness of Google when performing the DD steps, which help to better understand the main capabilities of web search engines to search for K. Finally, we identified K concepts on the web, which provide the highest benefit for practitioners when making design decisions. This can provide a guidance for practitioners to share their K on the web.

The results of this study motivate us to extend current K capturing approaches to focus on the most useful K sources (e.g. blogs), and the most scarce and useful K concepts (e.g. design decisions and drawbacks of solutions). Moreover, we aim to propose specialized web searching approaches to enhance the effectiveness of searching for K.

References

[1] L. Bass, P. Clements, and R. Kazman, *Software rchitecture in Practice*, 3rd ed. ddison-Wesley Professional, 2012.

[2] I. Gorton, R. Xu, Y. Yang, H. Liu, and G. Zheng, "Experiments in Curation: Towards Machine- ssisted Construction of Software rchitecture Knowledge Bases," in *IEEE/IFIP ICS 2017*, 4 2017, pp. 79–88.

[3] M. Bhat, K. Shumaiev, . Biesdorf, U. Hohenstein, and F. Matthes, " utomatic extraction of design decisions from issue management systems: machine learning based approach," in *Lecture Notes in Computer Science (including subseries Lecture Notes in rtificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10475 LNCS. Springer Verlag, 2017, pp. 138–154.

[4] M. Soliman, M. Riebisch, and U. Zdun, "Enriching rchitecture Knowledge with Technology Design Decisions," in *WICS* , 5 2015, pp. 135–144.

[5] O. Zimmermann, J. Koehler, F. Leymann, R. Polley, and N. Schuster, "Managing architectural decision models with dependency relations, integrity constraints, and production rules," *Journal of Systems and Software*, vol. 82, no. 8, pp. 1249–1267, 2009.

[6] . Tang, P. vgeriou, . Jansen, R. Capilla, and M. li Babar, " comparative study of architecture knowledge management tools," *Journal of Systems and Software*, vol. 83, no. 3, pp. 352–370, 2010. [Online]. vailable: http://www.sciencedirect.com/science/article/B6V0N-4X4GHP5-1/2/84a45c0d6dda12f7f563273ff85be120

[7] M. Soliman, M. Galster, and M. Riebisch, "Developing an Ontology for rchitecture Knowledge from Developer Communities," in *IEEE/IFIP ICS 2017*, 4 2017, pp. 89–92. [Online]. vailable: https://www.inf.uni-hamburg.de/en/inst/ab/swk/research/publications/pdf/2017-soliman-icsa.pdf

[8] . Barua, S. W. Thomas, and . E. Hassan, "What re Developers Talking bout? n nalysis of Topics and Trends in Stack Overflow," *Empirical Softw. Engg.*, vol. 19, no. 3, pp. 619–654, 6 2014.

[9] I. Gorton, J. Klein, and . Nurgaliev, " rchitecture Knowledge for Evaluating Scalable Databases," in *WICS , IEEE/IFIP*, 5 2015, pp. 95–104.

[10] R. Capilla, . Jansen, . Tang, P. vgeriou, and M. . Babar, "10 Years of Software rchitecture Knowledge Management," *J. Syst. Softw.*, vol. 116, no. C, pp. 191–205, 6 2016. [Online]. vailable: http://dx.doi.org/10.1016/j.jss.2015.08.054

[11] X. Xia, L. Bao, D. Lo, P. S. Kochhar, . E. Hassan, and Z. Xing, "What do developers search for on the web?" *Empirical Software Engineering*, vol. 22, no. 6, pp. 3149–3185, 12 2017. [Online]. vailable: https://link.springer.com/article/10.1007/s10664-017-9514-4

[12] R. Kazman and H. Cervantes, *Designing Software rchitectures: Practical pproach*. ddison-Wesley Professional, 2016.

[13] K. Byström and P. Hansen, "Conceptual framework for tasks in information studies," *Journal of the merican Society for Information Science and Technology*, vol. 56, no. 10, pp. 1050–1061, 2005. [Online]. vailable: http://dx.doi.org/10.1002/asi.20197

[14] P. Runeson, *Case study research in software engineering : guidelines and examples*. Wiley, 2012.

[15] K. Pearson, "I. Mathematical contributions to the theory of evolution. {\textemdash}{VII}. On the correlation of characters not quantitatively measurable," *Philosophical Transactions of the Royal Society of London. Series , Containing Papers of a Mathematical or Physical Character*, vol. 195, no. 262-273, pp. 1–47, 1 1900.

[16] M. Soliman, . Rekaby Salama, M. Galster, O. Zimmermann, and M. Riebisch, "Improving the Search for rchitecture Knowledge in Online Developer Communities," in *Proceedings - 2018 IEEE 15th International Conference on Software rchitecture, ICS 2018*. Institute of Electrical and Electronics Engineers Inc., 7 2018, pp. 186–195.

[17] O. Zimmermann, " rchitectural decision identification in architectural patterns." in *WICS /ECS Companion Volume*, ser. CM International Conference Proceeding Series, T. Männistö, M. . Babar, C. E. Cuesta, and J. E. Savolainen, Eds., vol. 704. CM, 2012, pp. 96–103.

[18] . Jansen and J. Bosch, "Software rchitecture as a Set of rchitectural Design Decisions," in *WICS* , 2005, pp. 109–120.

[19] J. Yin and J. Wang, " Dirichlet multinomial mixture model-based approach for short text clustering," in *Proceedings of the CM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, New York, US : ssociation for Computing Machinery, 2014, pp. 233–242. [Online]. vailable: http://dl.acm.org/citation.cfm?doid=2623330.2623715

[20] C. D. Manning, P. Raghavan, and H. Schütze, "Introduction to Information Retrieval," 2008. [Online]. vailable: https://nlp.stanford.edu/IR-book/information-retrieval-book.html

[21] S. Gottipati, D. Lo, and J. Jiang, "Finding Relevant nswers in Software Forums," in *IEEE/ CM SE 2011*, ser. SE '11. IEEE Computer Society, 2011, pp. 323–332.

[22] J. Cohen, "Statistical Power nalysis for the Behavioral Sciences — ScienceDirect." [Online]. vailable: https://www.sciencedirect.com/book/9780121790608/statistical-power-analysis-for-the-behavioral-sciences

[23] . Dean, D. Voss, and D. Draguljić, *Design and nalysis of Experiments*, ser. Springer Texts in Statistics. Cham: Springer International Publishing, 2017. [Online]. vailable: http://link.springer.com/10.1007/978-3-319-52250-0

[24] M. Soliman, M. Galster, . R. Salama, and M. Riebisch, " rchitectural Knowledge for Technology Decisions in Developer Communities: n Exploratory Study with StackOverflow," in *IEEE/IFIP WICS 2016*, 4 2016, pp. 128–133.

[25] P. Kruchten, P. Lago, and H. Vliet, "Building Up and Reasoning bout rchitectural Knowledge," in *Quality of Software rchitectures*, ser. Lecture Notes in Computer Science, C. Hofmeister, I. Crnkovic, and R. Reussner, Eds. Springer Berlin Heidelberg, 2006, vol. 4214, pp. 43–58.

[26] . Tang, Y. Jin, and J. Han, " rationale-based architecture model for design traceability and reasoning," *Journal of Systems and Software*, vol. 80, no. 6, pp. 918–934, 6 2007. [Online]. vailable: http://linkinghub.elsevier.com/retrieve/pii/S0164121206002287

[27] . El Malki and U. Zdun, "Guiding architectural decision making on service mesh based microservice architectures," in *Lecture Notes in Computer Science (including subseries Lecture Notes in rtificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11681 LNCS. Springer Verlag, 2019, pp. 3–19.

[28] S. Malakuti, T. Goldschmidt, and H. Koziolek, " catalogue of architectural decisions for designing IIoT systems," in *Lecture Notes in Computer Science (including subseries Lecture Notes in rtificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11048 LNCS. Springer Verlag, 2018, pp. 103–111.

[29] M. M. Rahman, J. Barson, S. Paul, J. Kayani, F. . Lois, S. F. Quezada, C. Parnin, K. T. Stolee, and B. Ray, "Evaluating how developers use general-purpose web-search for code retrieval," in *Proceedings - International Conference on Software Engineering*. New York, NY, US : IEEE Computer Society, 5 2018, pp. 465–475. [Online]. vailable: https://dl.acm.org/doi/10.1145/3196398.3196425

[30] F. Hassan, C. Bansal, N. Nagappan, T. Zimmermann, and . H. wadallah, " n empirical study of software exceptions in the field using search logs," in *International Symposium on Empirical Software Engineering and Measurement*. New York, NY, US : IEEE Computer Society, 10 2020, pp. 1–12. [Online]. vailable: https://dl.acm.org/doi/10.1145/3382494.3410692

[31] L. Wang, L. Fang, L. Wang, G. Li, B. Xie, and F. Yang, " PIExample: n effective web search based usage example recommendation system for java PIs," in *2011 26th IEEE/ CM International Conference on utomated Software Engineering, SE 2011, Proceedings*, 2011, pp. 592–595.

[32] R. Sirres, T. F. Bissyandé, D. Kim, D. Lo, J. Klein, K. Kim, and Y. L. Traon, " ugmenting and structuring user queries to support efficient free-form code search," *Empirical Software Engineering*, vol. 23, no. 5, pp. 2622–2654, 10 2018. [Online]. vailable: https://link.springer.com/article/10.1007/s10664-017-9544-y