An expert recommendation system for design decision making
Who should be involved in making a design decision?

Manoj Bhat*, Klym Shumaiev*, Kevin Koch*, Uwe Hohenstein†, Andreas Biesdorf† and Florian Matthes*
* *Technische Universität München, Boltzmannstr. 3, 85748 Garching, Germany*
{*manoj.mahabaleshwar,klym.shumaiev,kevin.koch,matthes*}@*tum.de*
†*Siemens AG - Corporate Technology, Otto-Hahn-Ring 6, München 81739, Germany*
{*uwe.hohenstein,andreas.biesdorf*}@*siemens.com*

***Abstract*—In large software engineering projects, designing software systems is a collaborative decision-making process where a group of architects and developers make design decisions on how to address design concerns by discussing alternative design solutions. For the decision-making process, involving appropriate individuals requires objectivity and awareness about their expertise. In this paper, we propose a novel expert recommendation system that identifies individuals who could be involved in tackling new design concerns in software engineering projects. The approach behind the proposed system addresses challenges such as identifying architectural skills, quantifying architectural expertise of architects and developers, and finally matching and recommending individuals with suitable expertise to discuss new design concerns. To validate our approach, a quantitative evaluation of the recommendation system was performed using design decisions from four software engineering projects. The evaluation not only indicates that individuals with architectural expertise can be identified for design decision making but also provides quantitative evidence for the existence of personal experience bias during the decision-making process.**

***Keywords*-Software architecture; Design decision making; Expert recommendation; Machine learning;**

I. Introduction

Software systems are designed by the people, for the people; which correlates to the fact that "major problems of [software design] work are not so much technological as sociological in nature" [1]. Over the past decades, exploring human aspects in software engineering has been a prominent area of research [2]. Especially, identifying and onboarding people with relevant expertise to address stakeholders' concerns during the software development phases including design, development, and maintenance is critical to the success of *any* software project [3]. To aid stakeholders during the software development lifecycle, a special class of applications called recommendation systems in software engineering was developed. These applications often rely on data mining of large-scale software engineering data, the creation of the expert profiles, generation of usage models, and predicting a list of items to address specific concerns [4]. These concerns, for instance, include *which requirements to implement in the next software release?* [5], *which software components to reuse?* [6], and *which stakeholders should participate in the upcoming software project?* [7].

In the context of architectural knowledge management (AKM), recommendations to support the design decision-making (DDM) process are beneficial for architects and developers. Software architecture is considered as a set of architectural design decisions [8], [9] and architects regularly make design decisions to address stakeholders' concerns which affect the architectural elements of software systems [10]. These design decisions are made at different abstraction levels – high, medium, and realization-level decisions [11]. For instance, high-level decisions include the selection of architectural styles, medium-level decisions correspond to the selection of third-party software modules, and realization-level decisions include changes made to the source code of software systems. High-level and medium-level decisions have a higher impact on the design as compared to the realization-level decisions [12]. Furthermore, high-impact decisions are typically made in groups [13], wherein a team of architects and developers discuss the potential solutions to address design concerns and then proceed to implement and evaluate the corresponding decisions. In this context, it is beneficial to identify appropriate experts who should be involved in the DDM process.

Additionally, during the DDM process, architects and developers – more often than not – favor naturalistic-approach to decision making [14]–[16]. In this regard, experience and knowledge of architects and developers play a crucial role in the DDM process [16], [17]. Furthermore, empirical studies have shown that compared to novices, experienced architects and developers more effectively explore the problem and use efficient decision-making strategies [18]. However, in the software architecture domain, the quantification of architectural expertise is still an open challenge. This observation is supported by one of the discussion points at the International Conference on Software Architecture 2017 [19]:

> Identification of "architecture skills" within organizations and projects: Who or what are sources of architectural expertise and competencies in organizations and how can we identify them?...

This work has been (partially) funded by the German Ministry of Education and Research under grant number 01IS16043Q.

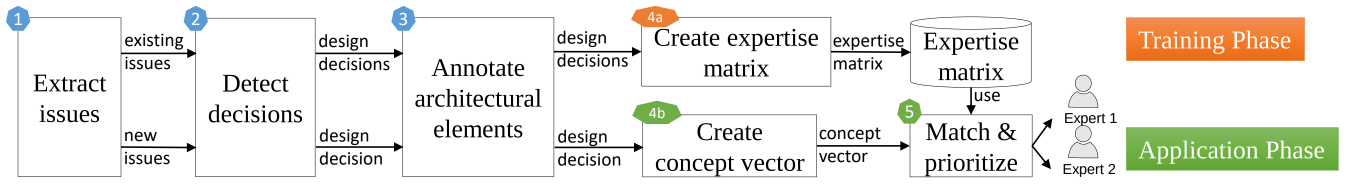


Figure 1. High-level overview of the expertise recommendation system

To address this challenge of identifying and quantifying architectural expertise of software architects and developers, we investigate design decisions extracted from issue management systems (IMS). It should be noted that even though design decisions are not explicitly documented, architects and developers implicitly capture decisions in tools such as IMS and version control systems [20]. We use IMS (for example, JIRA and GitHub issue tracker) as the data source within our approach for the following reasons:

- The use of IMS for managing system-related issues in agile software projects is becoming popular both in industry and open source communities [21]–[23].
- Medium- and realization-level design decisions can be automatically identified from issues with a high precision of 91.29% and a recall of 82.79% [24].

In this paper, we propose an expert recommendation system tailored to identify appropriate architects and developers to support the DDM process. The approach presented in this paper builds upon (a) a machine-learning based method to identify design decisions [24] and (b) an ontology-based approach to identify architectural elements [25]. Using these two approaches as a foundation, we realize a system for suggesting appropriate architects and developers for addressing design concerns. One of the benefits of this system is that practitioners do not have to manually document design decisions in an AKM tool. The system automatically extracts decisions from IMS, quantifies architectural expertise, and determines experts for new design concerns thereon.

Figure 1 illustrates the high-level overview of the proposed approach. During the **training phase**, first, all the existing issues of a project are imported into an AKM tool named AMELIE [26]. In Step 2, the decision detector component filters those issues that reflect decisions. In Step 3, architectural elements within the textual description of design decisions are automatically annotated (tagged) by the annotator component. For each design decision, the annotated elements and the individual who resolved the design decision are used to create an expertise matrix in Step 4a, which is then persisted for subsequent use.

During the **application phase**, a newly created issue goes through the same Steps 1, 2, and 3 as described above. A concept vector corresponding to the open design decision is computed in Step 4b. This vector is then compared against the expertise matrix in Step 5 to generate a list of experts who could be invited to discuss the design concern.

With this approach, we make an effort to quantify architectural expertise by analyzing decisions made by architects and developers in their past and ongoing projects with the help of an expertise matrix. This allows one to further search within the matrix to find people with relevant expertise so as to deal with specific concerns in a quantitative manner.

This paper is organized as follows. Section II presents the related work. In Section III, we elaborate the approach for recommending software architects and developers who could be involved in the design decision-making process. The evaluation of the recommendation system is presented in Section IV. In Section V, we discuss the lessons learned during the analysis of the concrete project datasets. Finally, we conclude with a short summary in Section VI.

II. RELATED WORK

In this section, we present the results of a semi-systematic literature review study. This study was conducted to investigate if any comparable research exists to support the DDM process with the aid of expert recommendation systems. We formulated a review protocol based on the systematic literature review guidelines [27]. This protocol specifies the background for the publication selection, search strategy, and data extraction and synthesis of the extracted data.

A. *Inclusion and exclusion criteria*

As an inclusion criterion for the selection process, we considered publications that recommended experts in the context of DDM process. We restricted ourselves to publications in the English language since we formulated the search queries using English terms and hence the results retrieved from the electronic databases were confined to the English language. To ensure that we retrieved all relevant studies over the years, we did not set a lower boundary on the publication date. The publications were retrieved on 12.12.2017 which is the upper boundary of the publication date. We excluded publications that did not explicitly relate to the involvement of experts during the DDM process. Furthermore, we also excluded all publications that did not provide any kind of tooling support or recommendation strategy for DDM.

B. *Search strategy*

As part of the search strategy, we queried the following electronic databases within the scope of this study:

- ACM Digital Library (http://portal.acm.org)
- IEEE Xplore (http://www.ieee.org/web/publications/xplore/)
- Science Direct (http://www.elsevier.com)

To ensure that all key publications were covered, we manually browsed through the following proceedings:

- International Conference on Software Architecture
- European Conference on Software Architecture
- Working IEEE Conference on Software Architecture

We formulated the search query in accordance with the inclusion criteria. That is, first we wanted to include publications in the area of "software architecture" or "software design". Second, we wanted to explore if there were any expert recommendation systems supporting the DDM process. To keep a broader search scope, we used "decision making" or "expert*" regular expressions as query Q2.

1) Q1: *"software architecture" OR "software design"*
2) Q2: *"decision making" OR "expert*"*
3) FSQ: *Q1 AND Q2*

The final search query (FSQ) means that at least one item from Q1 and Q2 should appear at least once in the publications. The search query was matched against the *title*, *keywords*, and *abstract* of publications in the database.

C. Data extraction and synthesis

The aforementioned search strategy leads to a total of 1,840 publications. The contents of these publications were assessed in four stages. In Stage 1, we retrieved the publications from the electronic databases. In Stage 2, first, we removed all duplicate papers. Second, (according to the exclusion criteria) after reading the title of the publications, all those publications that did not correlate to the topic of software architecture as well as to the DDM process were removed by the first author of this paper. This resulted in 407 publications. In Stage 3, the first author inspected the publications' abstract. All those publications that did not propose any tool support or recommendation strategy for the DDM process were removed. This finally resulted in 11 publications. These 11 publications were downloaded, read, and summarized by the second and third authors in Stage 4.

D. Summary report

A variety of approaches supporting different aspects of the DDM process were identified in the studied publications.

In [28], Stevanetic et al. proposed a tool structured around a central repository to support different stages of architecture documentation. During the exploratory, specification, and review stages, architects can search and choose facets (domain, tactic, patterns, and quality attributes) for their projects and the tool recommends design decisions corresponding to the selected facets. In [29], authors presented a tool that suggests alternative architectural decisions to architects for improving specific quality attributes of a software. On a similar note, the tool proposed by Silva et al. [30] recommends suitable architectural styles corresponding to the quality attributes of a system. Furthermore, given that one has to always make tradeoffs among quality attributes, Saadatmand and Tahvili [31] proposed a fuzzy decision support approach that identifies design alternatives which lead to the overall satisfaction of the quality attributes. Moaven et al. [32] formulated the architectural style selection as a multi-criteria decision-making problem and proposed a decision support system (DSS) based on fuzzy logic to identify suitable architectural styles for the system under consideration. Esfahaniet et al. [33] also presented the utility of fuzzy logic to help architects explore the solution space under uncertainty.

In [34], authors proposed a DSS which uses historical decisions captured in an ontology to analyze and reuse decisions in new scenarios. Another ontology-based approach was presented in [35] which allows architects to participate in the group decision-making process through a web interface. To support architects during the individual and group decision-making process, Tofan and Galster [36] presented a tool to capture and analyze architectural decisions.

Soliman et al. [37] conducted an exploratory study to understand how practitioners perceive technology solutions during the decision-making process and extended the existing architectural knowledge concepts to support architects in making a choice among different technology solutions. The technology-related architectural knowledge can be identified in StackOverflow posts which can be reused by architects during the technology decision-making process [38].

To summarize, various approaches for capturing, representing, and reasoning about architectural decisions have been proposed in the past. These approaches help architects to make more informed decisions while solving specific design problems. Unfortunately, none of the studied publications provide insights about recommendation systems for identifying architects who should be involved in addressing design problems. The approach presented in this paper complements the existing knowledge-based approaches by identifying relevant experts for the DDM process.

It should be noted that there exist some approaches especially targeting software defect management systems [3], [39]–[42]. These approaches identify developers who have expertise related to specific defects in software systems and automatically assign them to defect resolution. In our approach, we tailor the idea of representing expertise of individuals using an expertise matrix as presented in [3] to identify experts who should be involved in the DDM process. The main differences being that, the proposed system detects design decisions from issues, automatically extracts architectural topics, and then quantifies architectural expertise to recommend experts for engaging in the DDM process. Also, compared to document-clustering based approaches, we believe that the expertise matrix-based approach provides the flexibility to adapt the features of the documents to specific architectural topics. Moreover, as discussed in the subsequent sections, our approach provides more explicit and transparent results for the end-users.

III. Approach

In this section, we discuss the pipeline shown in Figure 1 that forms the core of the recommendation system. First, we introduce a machine-learning (ML) based approach to identify design decisions from issues in the issue management systems (IMS). Second, we briefly present an ontology-based approach to annotate text with architectural elements. Next, we discuss the expertise matrix used to capture the expertise profiles of software architects and developers. Finally, we present how we generate a concept vector for a new design concern and match it with the expertise matrix to recommend a list of experts.

A. Step 1: Extracting issues from issue management systems

To extract issues from IMS into our AKM system, we use an open source component named SyncPipes[1]. This extract-transform-load component not only helps us import existing issues but also synchronizes newly created issues with the data model of the AKM system.

B. Step 2: Automatically detecting design decisions

We use a supervised ML-based approach to automatically detect design decisions from issues captured in the IMS [24]. By using a linear support vector machine algorithm, design decisions can be detected with an accuracy (F1-score) of 91%. For training the ML model, two software architects created a dataset by manually labeling 1,571 issues from two large open source system (OSS) repositories. This publicly available dataset[2] contains 465 design decisions from the Apache Spark and 263 design decisions from the Apache Hadoop Common OSS projects. The trained ML model is also publicly available as part of the decision detector component[3]. This component labels an issue either as a "design decision" or "not a design decision" (Step 2 in Figure 1). If an issue is labeled as a design decision, then it is processed in the subsequent steps.

C. Step 3: Annotating architectural elements

To automatically identify and annotate (tag) natural language text with architectural elements, we use the broad cross-domain DBpedia ontology [43]. The DBpedia ontology comprises of concepts and relationships between concepts which are extracted from the Wikipedia articles. These concepts include architectural styles, patterns, software components, and technologies which correspond to the architectural elements in software systems. Using the DBpedia ontology, we can extract architectural elements from text with an accuracy (F1-score) of 84% [25]. For each issue labeled as design decision by the decision detector component, the annotator component extracts the architectural elements from its textual description and persists it in the AKM tool (Step 3 in Figure 1). The textual description of a design decision is derived from the *Summary* and *Description* attributes of the respective issue.

[1]https://wwwmatthes.in.tum.de/pages/2gh0u9d1afap
[2]https://wwwmatthes.in.tum.de/pages/9gvnwulxb4in
[3]https://github.com/sebischair/DocClassification

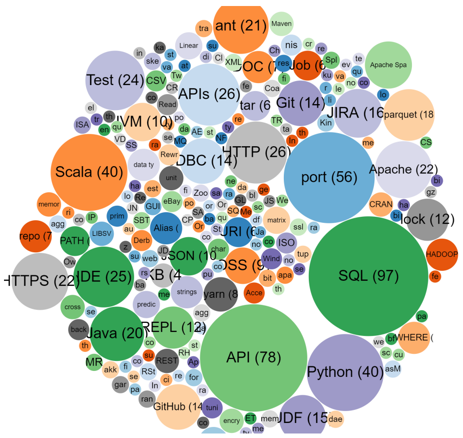


Figure 2. Annotated architectural elements for decisions in Spark dataset

Figure 2 shows the architectural elements and their occurrence counts for 465 design decisions in the Apache Spark project. For instance, API occurs 78 times, Scala 40 times, and HTTP 26 times. Each design decision is tagged with zero or more architectural elements. The occurrence count of an architectural element indicates the number of times this element has been discussed in individual design decisions. The annotated elements in design decisions are used to build the expertise matrix as discussed in the next section.

D. Step 4a: Building the expertise matrix

To quantitatively represent and measure the architectural expertise of architects and developers, we reuse the concepts from a previously proposed expert recommendation system. In particular, the terms *expertise atoms* and *expertise matrix* were introduced in [3]. With the expertise matrix, we represent individuals' expertise profiles as rows and architectural elements as columns. Let $D = \{d_1, d_2, d_3, ..., d_\mathrm{m}\}$ be the set of architects and developers, $E = \{e_1, e_2, e_3, ..., e_\mathrm{n}\}$ be the set of architectural elements, and V_{mn} be the expertise matrix. Here, m is the total number of architects and developers and n is the total number of architectural elements identified by the annotator component.

Expertise atoms (EAs) are the elementary atoms of expertise. They reflect an individual's expertise in a specific architectural topic. Each element $V[i][j]$ in V_{mn} represents an EA. By resolving a design concern, an individual gains expertise related to the corresponding architectural elements within the concern. The total count of the occurrences of an architectural element in all the design decisions resolved by an individual indicates his or her expertise level for the corresponding architectural element. That is, the higher the expertise level of an individual for an architectural element ($V[i][j]$) is, the more is the expertise in handling concerns related to the corresponding architectural element.

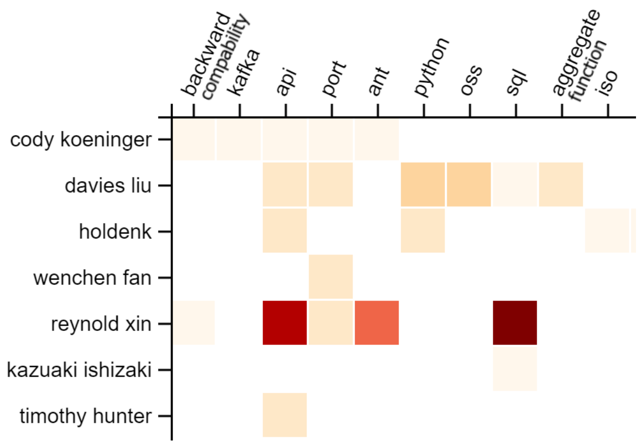


Figure 3. An excerpt of expertise matrix for Spark dataset

The expertise level of an individual for an architectural element can also be zero, which indicates that (s)he has not yet resolved a design concern pertaining to that architectural element and quantifies that (s)he has no expertise on that topic. Moreover, an individual can also have expertise in zero or more architectural elements.

Expertise profile: Each row ($V[i]$) within the expertise matrix (V_{mn}) represents an expertise profile for individual architects and developers. An individual's name is extracted from the "assignee" attribute of an issue, which represents the person who resolved this issue and gained expertise henceforth. Using this expertise profile, one can quantitatively assess the expertise of architects and developers corresponding to each architectural element.

Expertise matrix: Within this matrix, rows capture expertise profiles, columns represent architectural elements, and cells correspond to EAs. An excerpt of the expertise matrix for the Spark dataset is shown in Figure 3. The darker the color of a cell within the matrix, the higher the value for the corresponding EA. Once the matrix is generated for a project, it is persisted in the AKM tool and is further used to generate a list of experts for new design concerns.

E. Step 4b: Generating concept vectors

Once a new issue is added to the IMS and imported into the AKM tool, the decision detector component checks if it represents a design concern. If so, the annotator component identifies the architectural elements within this design concern. The frequency of occurrence of an architectural element represents its weighting factor within the concern. In essence, first, we identify those contributors who have expertise in one or more architectural elements within the design concern and then rank the individuals according to their expertise level corresponding to the weighting factor of architectural elements within the concern. Hence, to be able to compare the textual description of a design concern against the expertise matrix, we create an n-dimensional sparse vector, where n is the total number of identified architectural elements. We refer to this vector as the concept vector (CV) of a design concern. The CV is represented as a one-dimensional integer array of size n and each element in the array is initialized with zero values. The position of an architectural element in the CV is consistent with its column in the expertise matrix. The value in the position corresponding to each architectural element present in the new design concern is then replaced by its frequency count.

$$CV = \{c_1, c_2, c_3, ..., c_{\mathrm{n}}\}; where\ c_{\mathrm{i}} \geq 0;$$

F. Step 5: Generating the expert list

The CV computed in the previous step is matched against the expertise matrix stored in the AKM tool to generate the list of experts. The pseudocode for generating the expert list is described in Algorithm 1. The function "MATCH" takes as input the CV, expertise matrix, and the set of architects and developers. For each expertise profile (row) in the matrix, an expert vector (*EV*) of size n is created. Each element in *EV* is the product of the frequency of an architectural element ($CV[j]$) in a new design concern and the expertise level ($V[i][j]$) of the respective individual corresponding to that architectural element. For instance, if a new design concern emphasizes an architectural element with a higher frequency count and a specific individual has more expertise with that architectural element, then the score for that individual should proportionally increase with respect to both these variables. Once the EV is generated for an individual, we calculate the *score* as the magnitude (vector length) of that expert vector. The magnitude of the EV is calculated as the square root of the dot product of the vector by itself. Hence, the score generated for an individual is equally distributed across all architectural elements in the new design concern. If this score is greater than zero, the corresponding individual along with the score is added to the expert list. As the last step, after iterating over all expertise profiles, the expert list is ordered by the score.

Algorithm 1 Match and prioritize

```
function MATCH(CV, $V_{mn}$, D)
    $expertList \leftarrow \{\}$
    for i in 0..m do
        $expertVector \leftarrow newArray(n);$
        for j in 0..n do
            $EV[j] \leftarrow CV[j] \times V[i][j]$
        end for
                                        ▷ Compute score as vector magnitude
        $sum \leftarrow 0$
        for j in 0..n do
            $sum \leftarrow sum + EV[j] \times EV[j]$
        end for
        $score \leftarrow SQRT(sum)$
        if score >0 then
            $expertList.add(\text{“}person'', D[i])$
            $expertList.add(\text{“}score'', score)$
        end if
    end for
    $expertList \leftarrow ORDERBY(expertList, \text{“}score'')$
end function
```

ID	Name	Domain	Type	# design decisions	# unique contributors
1	Apache Spark	Data processing	Open source	447	95
2	Apache Hadoop Common	Distributed computing	Open source	238	111
3	Industry Project I	Connected Mobility	Closed source	368	13
4	Industry Project II	Knowledge management	Closed source	143	14

Table I
EVALUATION DATASET

IV. EVALUATION

In this section, we present the results of evaluating our approach using datasets from four different software projects which maintain issues in JIRA. Two of these projects are OSS projects and the other two projects are from our industry partner. As shown in Table I, the OSS projects have a higher number of unique contributors: the Apache Spark project has 95 unique contributors who resolved 447 design decisions and the Apache Hadoop Common project has 111 unique contributors who resolved 238 design decisions. The industrial projects are comparatively smaller, wherein, only 13 and 14 unique contributors resolved design decisions in Project I and II respectively.

For analyzing the results for individual datasets, we adhere to the following strategy:
Step 1: Order design decisions based on the *resolution date*.
Step 2: Split design decisions dataset into training [90% to 30%] and testing dataset [10% to 70%].
Step 3: Use the training dataset to create the matrix.
Step 4: For each design decision in the testing dataset, identify experts by matching the CVs against the matrix.
Step 5 - OSS projects: Measure the precision at 5, 10, 15, 20, 25, 30, and max. The precision at 5 (P@5) indicates if an individual who actually resolved the design decision belongs to the top 5 results in the list of recommended experts. Here, the *max* value refers to the total number of experts who can be recommended, that is, those individuals who resolved design decisions related to architectural elements *at least* once. Note that the recommended list is the list of top n experts in the context of P@n where n is the size of the list.
Step 6 - industry projects: Measure the precision at 2, 4, and 6. Since there are only 13 and 14 contributors in the industry projects, using a larger list will result in higher accuracy but will not lead to any interesting observations.

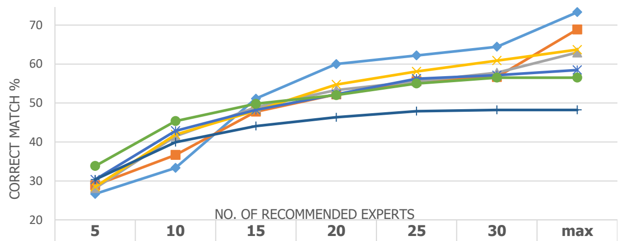
The overall accuracy of the algorithm can be calculated as the average of P@max across the investigated projects. As discussed in the subsequent subsections, even though the overall accuracy with P@max is higher than 60%, we believe that it does not provide useful insights; because the average P@max varies depending on the project. However, understanding the behavior with smaller recommendations (P@5 and P@10) and observing the trend across different list sizes (P@5 to P@max) and split strategies is interesting for researchers to reflect on the influence of project characteristics on the recommendation system and vice versa.

Furthermore, it should be noted that since design concerns were originally assigned to contributors without the aid of any system, the precision values should be interpreted as lower-bound estimates of the accuracy. By checking if an individual who actually resolved a design concern lies in the recommended expert list, we can only argue about the assumption that new design concerns are assigned to individuals who have dealt with similar cases in the past. In the subsequent subsections, we show that this assumption gets stronger as the size of the recommendation list increases.

A. Apache Spark

Among the datasets, the Apache Spark dataset has the largest number of labeled design decisions (447) with 95 unique contributors who resolved those design decisions. As shown in Figure 4, when we increase the size of the training dataset for creating the expertise matrix (from 60% to 90%) and the size of the recommendation list (see from P@15 to P@max), the precision values also increases. This is rather intuitive as one could imagine that when we increase the size of the training dataset as well as the solution space (expert list), the accuracy must also increase. However, if we observe the results for P@5 and P@10, using larger training datasets **decreases** the accuracy. The reason for this is that when we use a larger training dataset, the values of expertise atoms gets distributed across the corresponding expertise matrix and the individual who resolved a design decision in the testing dataset might not be present in the recommended list of more qualified personnel. On the contrary, when we use a smaller training dataset (40% and 50%), the expertise matrix is rather concise and results in higher accuracy (P@5 and P10) as compared to a larger training dataset. This is an important observation, since, for a 100 members team of architects and developers, we would not want to recommend more than 5 to 10 key experts who should be involved during the DDM process. Hence, it is necessary to consider an optimal size of the training dataset to prevent an overfitting of the matrix and to subsequently use it for recommendations. In case of the Spark project, using 40% or 50% of the dataset (approx. 200 design decisions) is sufficient to recommend experts for DDM.

Furthermore, reduced P@5 and P@10 values with a larger training dataset indicate that these design decisions are not made by a selected few individuals but is well distributed among the architects and developers. As it should be in an ideal case, this indicates a “healthy” project where knowledge does not reside only with a few experts.



	5	10	15	20	25	30	max
90%	26.67	33.33	51.11	60	62.22	64.44	73.33
80%	28.89	36.67	47.78	52.22	55.56	56.67	68.89
70%	28.15	41.48	48.89	53.33	55.56	57.78	62.96
60%	28.49	41.9	48.04	54.75	58.1	60.89	63.69
50%	30.36	42.86	48.21	52.23	56.25	57.14	58.48
40%	33.83	45.35	49.81	52.04	55.02	56.51	56.51
30%	30.35	39.94	44.09	46.33	47.92	48.24	48.24

Figure 4. Evaluation for Spark Dataset

B. Apache Hadoop Common

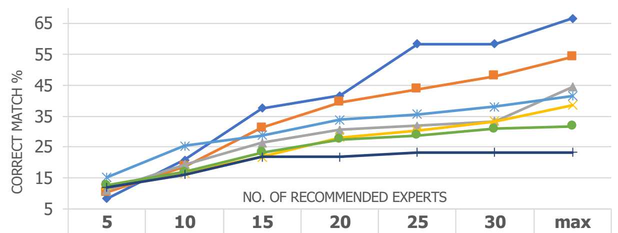
The Apache Hadoop Common dataset comprises of 238 design decisions which have been resolved by 111 unique contributors. As shown in Figure 5, we observe results similar to the results of the Apache Spark dataset.

- When we increase both the size of the training dataset (from 60% to 90%) and the recommendation list (from P@15 to P@max), the accuracy also increases.
- For P@5 and P@10, the smaller training dataset (50%) outperforms the larger dataset.
- Lower P@5 and P@10 values for larger training datasets indicate a "healthy" project where design decisions are not made by a few architects and developers.

Similar observations from both these OSS projects indicate that even a smaller design decisions dataset is sufficient to build a comprehensive expertise matrix to recommend experts who could be involved in the DDM process.

C. Industry Project I

Unlike the OSS projects wherein stakeholders have maintained issues in JIRA since 2012, the industrial projects are under development and maintenance since 2016 and the team size of architects and developers is considerably smaller. The first project under consideration aims to provide mobility-related services for commuters in metropolitan areas by benefiting from the sensor data collected from different means of transportation. Stakeholders of this project have captured 1,233 issues in JIRA. Using the decision detection model, we have identified 368 design decisions which have been resolved by 13 unique contributors. After creating the expertise matrix using the training dataset and matching the concept vectors of design decisions in the testing dataset, we measured the precision at 2, 4, and 6 (due to fewer contributors). Since this project has a large number of design decisions (368), as shown in Figure 6, the precision values do not vary significantly for different split strategies. Similar to the OSS projects, increasing the size of the training dataset and the size of the recommendation list also increases the accuracy of the recommendation system.



	5	10	15	20	25	30	max
90%	8,33	20,83	37,5	41,67	58,33	58,33	66,67
80%	10,42	18,75	31,25	39,58	43,75	47,92	54,17
70%	11,11	19,44	26,39	30,56	31,94	33,33	44,44
60%	12,5	16,67	21,88	28,13	30,21	33,33	38,54
50%	15,25	25,42	28,81	33,9	35,59	38,14	41,53
40%	12,68	16,9	23,24	27,46	28,87	30,99	31,69
30%	11,97	16,2	21,83	21,83	23,24	23,24	23,24

Figure 5. Evaluation for Hadoop Common dataset

Contrary to the results of the OSS projects, we observed higher accuracy for P@2 and P@4 in case of industrial projects (cf. Figure 6 and 7). The average accuracy (P@2 and P@4) across different splits is 44.38% and 71.32% respectively. That is, in 44% of the cases, either of the two individuals who had the most expertise actually resolved the design decision. Similarly, 71% of decisions were resolved by the top four individuals with most expertise. Furthermore, surprisingly, either one of the top two recommended experts remained consistently in all the recommendations.

Even though the higher accuracy indicates that the system can successfully identify experts who can deal with specific design decisions, the fact that only those individuals actually resolved most of the decisions is not "healthy" for the project. It shows that there are only a few individuals with relevant expertise and the chances of knowledge vaporization in case they leave the project is higher.

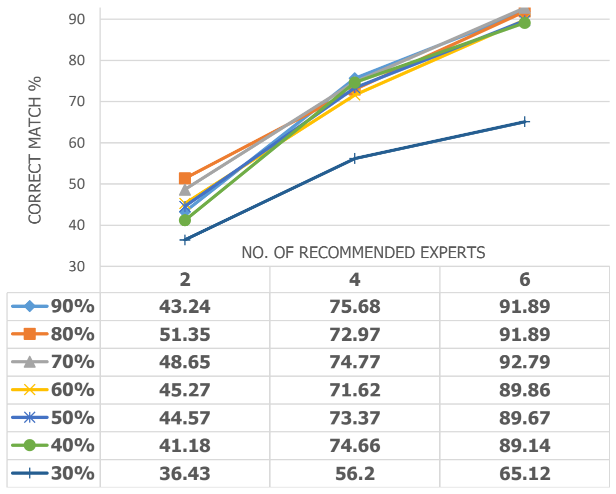
D. Industry Project II

The second industry project that we analyzed is a knowledge management system which guides stakeholders during different phases of the application lifecycle of software projects. Stakeholders of this project have maintained 1,153 issues in JIRA since early 2016 and the decision detection model identified 143 design decisions which were resolved by 14 unique contributors. As shown in Figure 7, the results are similar to that of the first industry project:

- The average accuracy of finding experts in the top 2 and top 4 recommendation list is as high as 58.60% and 73.77% respectively.
- Higher accuracy (P@2 and P@4) and consistently recommending either one of the top two experts indicate that decisions were made only by a few individuals and there is a need for knowledge transfer within the team.

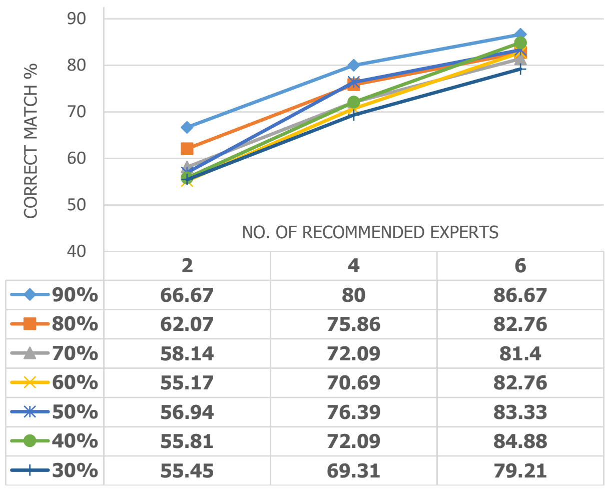
V. LESSONS LEARNED AND FUTURE WORK

In this section, we share our experiences and lessons learned during the analysis of the project datasets as well as discuss the feedback from our industry partner.



	2	4	6
90%	43.24	75.68	91.89
80%	51.35	72.97	91.89
70%	48.65	74.77	92.79
60%	45.27	71.62	89.86
50%	44.57	73.37	89.67
40%	41.18	74.66	89.14
30%	36.43	56.2	65.12

Figure 6. Evaluation for Industry Project I dataset



	2	4	6
90%	66.67	80	86.67
80%	62.07	75.86	82.76
70%	58.14	72.09	81.4
60%	55.17	70.69	82.76
50%	56.94	76.39	83.33
40%	55.81	72.09	84.88
30%	55.45	69.31	79.21

Figure 7. Evaluation for Industry Project II dataset

A. *Project and team characteristics*

As presented in Section IV, we observed higher accuracy with smaller recommendation lists for the industry projects as compared to the OSS projects. This is due to a couple of factors, namely, the number of contributors and the culture of assignment of issues. Typically, teams of small and medium-sized projects (with 10 to 50 architects and developers) contain only a few architects and senior developers. The key design decisions including setting up the IT infrastructure, selecting the communication protocol, and making changes to the data models are made by those experienced architects and developers. Hence, for new design decisions, the proposed algorithm correctly identifies those experts in such smaller teams as compared to larger teams where design decisions are resolved by many contributors. Furthermore, in the OSS projects, contributors independently resolve issues (bottom-up) by submitting a pull request in the code repository which is then merged into the main branch and the respective issue is closed. Whereas, in the industry projects we analyzed, issues are assigned to developers by architects or senior developers (top-down). Hence, contrary to the industry projects, in the OSS projects where developers had more freedom to choose the design problems, we observed that the values of the expertise atoms were scattered across the expertise matrix. To avoid the risk of knowledge vaporization, we prescribe the use of the expertise matrix to identify hotspots within the matrix (rows containing darker cells) early in the project so as to ensure the involvement of junior developers while addressing design concerns.

Within the scope of this study, we have not addressed the cold start problem in the recommendation system. That is, when new contributors join the team, it is not possible to create their expertise profiles. However, we consider this to be a technical challenge as one could integrate external data sources such as professional resumes to extract their skills.

Another shortcoming of our approach is that we have to consider an optimal size of the training dataset for creating the expertise matrix. As discussed in Section IV, using a larger training dataset might not result in better accuracy. The size of the training dataset has to be dealt on a project-to-project basis. Within the scope of this study, we could not generalize the optimal size of the training dataset.

One of the frequent concerns raised by our industry partner is "how to deal with issues captured in different languages". In some projects, either all the issues are in the German language or there is a mixture of both the English and the German text. Dealing with such scenarios is challenging as we not only have to use a translation service but also retrain the decision detection models which is time and effort intensive.

Finally, during the demonstration of the results to the stakeholders of the Industry Project II, one of the architects expressed that it would be beneficial to apply the approach across the organization's projects so as to know with whom one has discuss for resolving similar design problems. Addressing this point is not a technical problem but it is difficult since every project within an organization has different confidentiality criteria. Moreover, as also pointed out by one of the stakeholders, not every individual is comfortable with the idea that their architectural expertise is being quantified.

B. *Expert recommendation*

The architects from the Industry Project I highlighted that the system should also consider attributes including availability and workload of experts as well as criticality and priority of design decisions. Since projects maintain such structured information in issue management systems such as JIRA, we can consider these complimentary parameters while generating the list of experts.

Furthermore, it is necessary to emphasize that the aim is not to automatically assign contributors to address a design concern but to recommend a list of experts who should be involved in the DDM process. In this context, first, we need to have a balanced mix of both senior and junior architects and developers in the list so as to ensure knowledge transfer. Second, it is not sufficient to present only the list of experts but we also need to identify and assign roles for the experts such as owners, decision-makers, and moderators.

We are currently investigating the aforementioned aspects of improving the quality of the recommendation list as part of a follow-up research project[4].

C. Personal experience and cognitive biases

In our approach, we make the assumption that architects and developers – intentionally or unintentionally – rely on their past experiences while making design decisions or even when selecting a design problem to be addressed. The use of experience as an "anchor" while making decisions may lead to **anchor** and **confirmation** cognitive biases [44]. The qualitative interview-based studies [13], [18] have indicated personal experience to be a key factor influencing the DDM process. The recommendation results for the OSS projects show that there is a high chance (cf. P@max; avg. 61.72% for Spark and 42.89% for Hadoop) that contributors select similar concerns that they have addressed in the past. Similarly, in the industry projects, issues reflecting design decisions are assigned to those developers who have dealt with similar concerns in the past (cf. P@6; avg. 91.22% for Industry Project I and 82.45% for Project II).

The aforementioned observation provides quantitative evidence to indicate that experience of architects and developers play an important role when they select and resolve design decisions which in turn influences the DDM process.

VI. CONCLUSIONS

Architectural design decisions have a long-lasting impact on the architecture of software systems. These decisions have to be made carefully by experienced architects and developers, or at least they must be consulted before implementing those design decisions. In this paper, we proposed an automatic approach to address the challenge of identifying experts who should be involved in the DDM process. Within this approach, we tackled the challenges of identifying architectural elements within projects, quantifying architectural expertise of architects and developers corresponding to specific skills, and finally matching and recommending individuals with suitable expertise to discuss new design concerns. Furthermore, this approach is not only applicable for recommending experts who should be involved in addressing design concerns, but is also applicable for identifying experts who can fix a given issue in IMS.

The proposed recommendation system was evaluated using datasets from four software projects with different project characteristics. Apart from the observation that the system successfully identifies experts, we also observed that even a smaller training dataset of design decisions is sufficient for creating an appropriate expertise matrix. However, to create an expertise matrix we have to optimize the size of the training dataset which can only be decided from project-by-project or team-by-team basis depending on how people are assigned to projects in specific organizations. As a byproduct of our analysis, we identified two important aspects. First, we can use the expertise matrix to identify hotspots to observe the health (with respect to architectural knowledge distribution) of a project. If there are many rows (expertise profiles) with sparse cells (no values in expertise atoms) and few rows with dense cells, strategies should be defined for knowledge transfer to avoid knowledge evaporation. Second, the evaluation results provide quantitative evidence for the existence of personal experience bias when architects and developers address design concerns.

Finally, our work should be considered as the first steps towards the realization of an expert recommendation system to support the DDM process. We plan to address the shortcomings of our approach as discussed in Section V in the next iteration. As initial steps, we are working on gathering feedback from our industry partner to ensure (a) comprising a well-mixed list of both experts and novices, (b) defining and assigning roles to experts within the expert list, and (c) considering time constraints, availability, and workload of experts while creating the expert list.

[4]https://wwwmatthes.in.tum.de/pages/1v8023c2kodm1

REFERENCES

[1] T. DeMarco and T. Lister, *Peopleware: productive projects and teams*. Addison-Wesley, 2013.

[2] P. Lenberg, R. Feldt, and L. G. Wallgren, "Behavioral software engineering: a definition and systematic literature review," *J. Syst. and Soft.*, vol. 107, pp. 15–37, 2015.

[3] A. Mockus and J. D. Herbsleb, "Expertise browser: a quantitative approach to identifying expertise," in *Proc. 24th Int. Conf. on Soft. Eng.* ACM, 2002, pp. 503–512.

[4] M. P. Robillard, W. Maalej, R. J. Walker, and T. Zimmermann, *Recommendation systems in software engineering*. Springer Science & Business, 2014.

[5] A. Felfernig, C. Zehentner, G. Ninaus, H. Grabner, W. Maalej, D. Pagano, L. Weninger, and F. Reinfrank, "Group decision support for requirements negotiation." in *UMAP Workshops*. Springer, 2011, pp. 105–116.

[6] F. McCarey, M. Ó. Cinnéide, and N. Kushmerick, "Rascal: a recommender agent for agile reuse," *Artificial Intelligence Review*, vol. 24, pp. 253–276, 2005.

[7] S. L. Lim, D. Quercia, and A. Finkelstein, "Stakenet: using social networks to analyse the stakeholders of large-scale software projects," in *Proc. 32nd ACM/IEEE Int. Conf. on Soft. Eng.-Volume 1.* ACM, 2010, pp. 295–304.

[8] P. Kruchten, "An ontology of architectural design decisions in software intensive systems," in *2nd Groningen Workshop on Soft. Variability*. Groningen, The Netherlands, 2004, pp. 54–61.

[9] A. Jansen and J. Bosch, "Software architecture as a set of architectural design decisions," in *5th Working IEEE/IFIP Conf. on Soft. Architecture*. IEEE, 2005, pp. 109–120.

[10] M. ISO, "Systems and software engineering–architecture description," ISO/IEC/IEEE 42010, Tech. Rep., 2011.

[11] J. S. van der Ven and J. Bosch, "Making the right decision: supporting architects with design decision data," in *European Conf. on Soft. Architecture*. Springer, 2013, pp. 176–183.

[12] A. G. J. Jansen, "Architectural design decisions," 2008.

[13] I. Groher and R. Weinreich, "A study on architectural decision-making in context," in *12th Working IEEE/IFIP Conf. on Soft. Architecture*. IEEE, 2015, pp. 11–20.

[14] C. Zannier, M. Chiasson, and F. Maurer, "A model of design decision making based on empirical results of interviews with software designers," *Inform. and Soft. Technol.*, vol. 49, no. 6, pp. 637–653, 2007.

[15] S. T. Hassard, A. Blandford, and A. L. Cox, "Analogies in design decision-making," in *Proc. 23rd British HCI Group Annual Conf. on People and Computers: Celebrating People and Technol.* British Computer Society, 2009, pp. 140–148.

[16] C. Zannier and F. Maurer, "Social factors relevant to capturing design decisions," in *Proc. 2nd Workshop on SHAring and Reusing Architectural Knowledge Architecture, Rationale, and Design Intent*. IEEE Computer Society, 2007, p. 1.

[17] H. van Vliet and A. Tang, "Decision making in software architecture," *J. Syst. and Soft.*, vol. 117, pp. 638–644, 2016.

[18] A. Tang, M. Razavian, B. Paech, and T.-M. Hesse, "Human aspects in software architecture decision making: a literature review," in *IEEE Int. Conf. on Soft. Architecture*. IEEE, 2017, pp. 107–116.

[19] M. Galster, D. A. Tamburri, and R. Kazman, "Towards understanding the social and organizational dimensions of software architecting," *ACM SIGSOFT Soft. Eng. Notes*, vol. 42, no. 3, pp. 24–25, 2017.

[20] C. Miesbauer and R. Weinreich, "Classification of design decisions–an expert survey in practice," in *European Conf. on Soft. Architecture*. Springer, 2013, pp. 130–145.

[21] G. Antoniol, K. Ayari, M. Di Penta, F. Khomh, and Y.-G. Guéhéneuc, "Is it a bug or an enhancement?: a text-based approach to classify change requests," in *Proc. 2008 Conf. Center for Advanced Studies on Collaborative Research: Meeting of Minds*. ACM, 2008, p. 23.

[22] G. Goth, "Agile tool market growing with the philosophy," *IEEE Software*, vol. 26, no. 2, pp. 88–91, 2009.

[23] U. Van Heesch, T. Theunissen, O. Zimmermann, and U. Zdun, "Software specification and documentation in continuous software development: a focus group report," in *Proc. 22nd European Conf. on Pattern Languages of Programs*, ser. EuroPLoP '17. ACM, 2017, pp. 1–13.

[24] M. Bhat, K. Shumaiev, A. Biesdorf, U. Hohenstein, and F. Matthes, "Automatic extraction of design decisions from issue management systems: a machine learning based approach," in *European Conf. on Soft. Architecture*. Springer, 2017, pp. 138–154.

[25] M. Bhat, K. Shumaiev, A. Biesdorf, U. Hohenstein, M. Hassel, and F. Matthes, "An ontology-based approach for software architecture recommendations," in *23rd Americas Conf. on Inform. Syst., AMCIS 2017, Boston, MA, USA, August 10-12, 2017*, 2017.

[26] M. Bhat, K. Shumaiev, A. Biesdorf, U. Hohenstein, and F. Matthes, "Meta-model based framework for architectural knowledge management," in *Proc. of the 10th European Conf. on Soft. Architecture Workshops*. ACM, 2016, p. 12.

[27] D. Budgen and P. Brereton, "Performing systematic literature reviews in software engineering," in *Proc. 28th Int. Conf. on Soft. Eng.* ACM, 2006, pp. 1051–1052.

[28] S. Stevanetic, K. Plakidas, T. B. Ionescu, F. Li, D. Schall, and U. Zdun, "Tool support for the architectural design decisions in software ecosystems," in *Proc. 9th European Conf. on Soft. Architecture Workshops*. ACM, 2015, p. 45.

[29] D. Ameller, O. Collell, and X. Franch, "Architech: tool support for nfr-guided architectural decision-making," in *Proc. 20th Int. Requirements Eng. Conf.* IEEE, 2012, pp. 315–316.

[30] I. C. Lopes Silva, P. H. Brito, B. F. dos S Neto, E. Costa, and A. A. Silva, "A decision-making tool to support architectural designs based on quality attributes," in *Proc. 30th Annual ACM Symposium on Applied Computing*. ACM, 2015, pp. 1457–1463.

[31] M. Saadatmand and S. Tahvili, "A fuzzy decision support approach for model-based tradeoff analysis of non-functional requirements," in *Proc. 12th Int. Conf. on Inform. Technol.-New Generations*. IEEE, 2015, pp. 112–121.

[32] S. Moaven, J. Habibi, H. Ahmadi, and A. Kamandi, "A decision support system for software architecture-style selection," in *Proc. 6th Int. Conf. on Soft. Eng. Res., Manage. and Appl.* IEEE, 2008, pp. 213–220.

[33] N. Esfahani, S. Malek, and K. Razavi, "Guidearch: guiding the exploration of architectural solution space under uncertainty," in *Proc. 35th Int. Conf. on Soft. Eng.* IEEE, 2013, pp. 43–52.

[34] A. Cicchetti, M. Borg, S. Sentilles, K. Wnuk, J. Carlson, and E. Papatheocharous, "Towards software assets origin selection supported by a knowledge repository," in *1st Int. Workshop on Decision Making in Soft. ARCHitecture*. IEEE, 2016, pp. 22–29.

[35] J. Chai and J. N. Liu, "An ontology-driven framework for supporting complex decision process," in *World Automation Congress*. IEEE, 2010, pp. 1–6.

[36] D. Tofan and M. Galster, "Capturing and making architectural decisions: an open source online tool," in *Proc. 8th European Conf. on Soft. Architecture Workshops*. ACM, 2014, p. 33.

[37] M. Soliman, M. Riebisch, and U. Zdun, "Enriching architecture knowledge with technology design decisions," in *12th Working IEEE/IFIP Conf. on Soft. Architecture*. IEEE, 2015, pp. 135–144.

[38] M. Soliman, M. Galster, A. R. Salama, and M. Riebisch, "Architectural knowledge for technology decisions in developer communities: An exploratory study with stackoverflow," in *13th Working IEEE/IFIP Conf. on Soft. Architecture*. IEEE, 2016, pp. 128–133.

[39] D. W. McDonald and M. S. Ackerman, "Expertise recommender: a flexible recommendation system and architecture," in *Proceedings of the 2000 ACM conference on Computer supported cooperative work*. ACM, 2000, pp. 231–240.

[40] J. Anvik, L. Hiew, and G. C. Murphy, "Who should fix this bug?" in *Proc. 28th Int. Conf. on Soft. Eng.* ACM, 2006, pp. 361–370.

[41] S. Minto and G. C. Murphy, "Recommending emergent teams," in *Proc. 4th Int. Workshop on Mining Soft. Repositories. ICSE Workshops MSR'07*. IEEE, 2007, pp. 5–5.

[42] J. Anvik and G. C. Murphy, "Reducing the effort of bug report triage: Recommenders for development-oriented decisions," *ACM Trans. on Soft. Eng. and Methodology*, vol. 20, no. 3, p. 10, 2011.

[43] P. N. Mendes, M. Jakob, and C. Bizer, "Dbpedia: A multilingual cross-domain knowledge base," in *LREC*, 2012, pp. 1813–1817.

[44] M. Razavian, A. Tang, R. Capilla, and P. Lago, "In two minds: how reflections influence software design thinking," *J. Soft.: Evolution and Process*, vol. 28, no. 6, pp. 394–426, 2016.