Architectural Knowledge for Technology Decisions in Developer Communities

An Exploratory Study with StackOverflow

Mohamed Soliman*, Matthias Galster†, Amr R. Salama*, and Matthias Riebisch*
*Department of Informatics, University of Hamburg, Germany
Email: {soliman, salama, riebisch}@informatik.uni-hamburg.de
† University of Canterbury, Christchurch, New Zealand
Email: mgalster@ieee.org

***Abstract*—Architectural decisions have a big influence on basic properties of a software system, and are difficult to change once implemented. Technology decisions deserve special attention because they are one of the most frequently occurring types of architectural decisions. Architectural knowledge is crucial for good decisions. Current architecture knowledge management approaches try to support architects by offering a rich base of architectural solutions and design decision rules. However, they mostly depend on manually capturing and maintaining the architectural knowledge. In this paper, we utilize the most popular online software development community (StackOverflow) as a source of knowledge for technology decisions to support architecture knowledge management approaches with a more efficient methods for knowledge capturing. We conducted an exploratory study, and followed a qualitative and quantitative content analysis approach. We analysed the posts in this community to identify architecture-relevant and technology-related knowledge, and to classify the posts into different types for the purpose of knowledge structuring. In addition, we evaluated our findings through feedback from practitioners.**

I. Introduction

Software architecture is about making design decisions which potentially impose risk on product success. Such decisions significantly affect structural properties of a system but also its quality attributes. Once an architectural solution is implemented it is quite difficult to change [1]. This differentiates software architecting from implementation e.g., fixing a bug in a Java method tends to be less complex than changing a technology or pattern.

Architecture Knowledge (AK) plays an important role in taking the right *Architectural Design Decisions (ADDs)* [2]. Some existing AK management approaches (e.g., [3], [4]) focus on building repositories of AK to guide architects during the decision making process. The repositories are populated with knowledge by architects manually, which is a time consuming and tedious process. As a result, the amount of gathered knowledge is limited. Moreover, their maintenance and evolution is an additional laborious task. Therefore, efficient methods for capturing and maintaining AK are needed to support existing knowledge management approaches.

ADDs could be classified into conceptual and technology ADDs [3]. Conceptual ADDs influence the system architecture configuration, independent of the implementation. Examples for conceptual ADDs are selecting architectural patterns and tactics [1], [5]. Technology ADDs are concerned with the selection of concrete technology solutions to implement the architecture, such as frameworks. The selection between technology solutions depends on architecture-significant technology aspects associated with technology features. These aspects differentiate technologies based on benefits and drawbacks [6]. A recent survey shows that executive decisions constitute around 25% of a system ADDs, and most of them are technology ADDs. In addition, technology ADDs are the mostly documented ADDs [7].

Social software (e.g., forums) offer new methods for knowledge capturing and sharing [8]. Software engineers can now ask questions, and learn from others through many technical social media. In our previous research we interviewed software architects about the process of technology design decision making [6]. Many architects mentioned social software as a source of knowledge for technology solutions. In addition, recent studies (e.g., [9]) on social media in software development show that developers use social media to discuss high-level concepts, such as features and domain concepts (which are architecture-relevant). These findings indicate that AK may exist in software development communities. Our main goal is to complement AK management systems with efficient methods for technology-related AK capturing, through utilizing existing developer communities as a source for this type of knowledge.

To achieve this, we need an example of online software development community that provides a good source of well-structured and architecturally related information. We selected *StackOverflow (SO)*[1]. SO is one of the biggest software development social media websites, which follows a Question and Answer (Q&A) structure. In addition, it supports useful knowledge management features, such as inclusion of context details in posts, the quality of the knowledge on SO is ensured through evaluation of posts from users, and the ability to manually categorize posts through assigning tags to them. Moreover, SO provides a very large pool of knowledge and information that is constantly updated with new posts from users. All these features could support AK management systems with valuable and evolvable technology-related AK, which has already been evaluated by technology experts.

In detail, we conduct an empirical study with the objective

[1]http://stackoverflow.com/

of identifying SO posts which provide useful technology-related AK. Furthermore, we explore the key differences between this type of SO posts and other programming posts. Based on this objective, we formulated our research questions:

- RQ1: What are the types of architecture-relevant posts (ARPs) in SO and how could we classify ARPs?
- RQ2: Which types of ARPs in SO do practitioners consider architecture-relevant?
- RQ3: What are terms in SO posts that distinguish ARPs from programming posts?

By answering the mentioned research questions, we create a comprehensive overview of the SO ARPs through understanding their types and classification. This overview is a framework for further analysis steps, such that each of the ARP types embodies different AK concepts (e.g., quality attributes, design decisions). If we identify the AK concepts within each ARP type, we can enable more approaches for AK capturing (e.g., automated mining and classification of ARP, as well as extraction of AK from SO).

II. Research Process

Our research process is divided into four phases, which are described in the following subsections.

A. Phase 1: Prepare StackOverflow Posts for Analysis

1) Phase 1a: Query for Candidate Posts (Sampling): SO posts could be classified into different topics. Some of these topics are architecture-related (e.g., posts about web development). On the other hand, posts about coding styles is an example for a topic which is programming-related. We selected the middleware topic as our sample topic for two reasons: 1) Middleware is an important topic in the software architecture field, for example, many architecture patterns (e.g., [5]) address interoperability issues. 2) The amount of middleware posts in SO is manageable for our exploratory study.

In order to gather candidate architecture-relevant middleware posts, we applied the following selection criteria: 1) Posts had to be concerned with at least one of 52 different middleware technologies (e.g., RabbitMQ, WCF). We queried post title, questions and tags for middleware technologies. We identified these technologies through a review of existing technologies on Wikipedia[2]. 2) We excluded posts with no answer. 3) We considered posts with a question score higher than or equal 7. This was to ensure the quality of the selected posts [10]. 4) We excluded posts which include blocks of source code in the question because most of those posts discuss programming problems [10]. The gathering process has been done using a set of SQL queries through the stack exchange explorer[3] and resulted in 2,561 posts.

2) Phase 1b: Initial Manual Classification of ARPs and Programming Posts: Two of the authors classified the same sample posts manually to differentiate architecture-relevant posts from programming posts. We conducted three iterations, including inter-coder reliability tests to ensure that the classifications of the two researchers were consistent and calculated the kappa coefficient [11] to determine whether agreement between researchers was beyond chance. In each iteration, 100 posts were randomly selected and classified by two researchers, different types of posts and examples were gathered, and disagreements were discussed. By the end of these iterations, we defined architecture-relevant posts, together with a set of classified post examples, according to the prototype theory of definition [12]. Based on this definition, we classified the rest of our sample as ARPs and programming posts. This step resulted in 858 ARPs, and 1,653 programming posts, while 50 posts have been excluded for being out of middleware scope.

[2]https://en.wikipedia.org/wiki/Category:Message-oriented_middleware
https://en.wikipedia.org/wiki/Category:Enterprise_application_integration
[3]https://data.stackexchange.com/stackoverflow/query/new

TABLE I: Background of participants

ID	IT Exp. (Years)	Software Arch. Exp. (Years)	Technology Background	Industries	Role
1	11	4	Microsoft Technologies	NLP, E-Commerce	Technology Consultant
2	10	4	Java / J2EE	Telecom, Billing, Defense	Enterprise Architect
3	11	5	C++, WebLogic, J2EE, Spring	Telecom	Technology Consultant
4	14	7	Microsoft Technologies	eGov, Financial	Solution Architect
5	17	8	Microsoft Technologies	Healthcare, Manufacturing	Program Manager
6	10	4	Java J2EE	Healthcare, Retail, Billing	Technology Consultant
7	11	5	J2EE, Weblogic, Unix, RabbitMQ	Telecom	System Engineer
8	12	4	Microsoft Technologies	Transportation	Technology Consultant
9	30	12	VME Mainframe, J2EE, Unix	Defense, Manufacturing	Solution Architect
10	13	4	Microsoft Technologies	Telecom	Solution Architect
11	6	1	Microsoft Technologies	Retail, Procurement	Developer

B. Phase 2: Classification of ARPs

In order to answer RQ1 and to classify the ARPs into types based on the concerns that they address, we followed a summarizing qualitative content analysis method [12], where a short summary about each post is written, and based on the summary, the post is assigned to a category. We applied this method to the gathered 858 candidate ARPs from phase 1b. The identified categories are presented in Section III as our ARP types.

C. Phase 3: Obtain Feedback from Practitioners

1) Phase 3a: Practitioners Selection: Table I shows the participants. All of the participants are familiar with SO and use it as part of their work. All participants work or worked in multinational companies with more than 100,000 employees.

2) Phase 3b: Evaluation Posts Sampling: In order to make the sample of posts given to practitioners representative to our overall sample of 2,561 posts, we conducted stratified random sampling [13] among the ARP types (phase 2), as well as the programming posts. This way of sampling guaranteed that all types of posts were included in the evaluation sample proportional to their occurrence in the overall sample.

3) Phase 3c: Evaluation Execution: Each participant received a sample of posts (see Table IV). The total number of posts used was 1,173 posts. It took each participant between 3 and 4 hours to classify the posts between ARPs and programming posts. The analysis of the feedback of the participants helped us to answer RQ2, which is presented in Section IV.

D. Phase 4: Identification of Terms in ARPs

In order to answer RQ3, we performed quantitative content analysis using a data mining algorithm based on the classified posts from phase 1.

1) Phase 4a: Preprocessing of Posts: We first removed stopwords, which are common words in the English language (e.g., "the", "is"). We also removed numbers and punctuation symbols. Then, we transformed each post section (title, question, and answers) into a "vector of features", which is a vector of 0s and 1s, where each element of a vector shows if a certain word of the list of all words in all posts exist in this post (value of 1) or not (value of 0). We considered the post title, question and answers separately. Thus, each post is associated with three vectors of features which represent the existence of words in each of the post sections. In addition, each vector is marked as being either an ARP or programming post.

2) Phase 4b: Terms Selection and Ranking: We used the "Information Gain Ratio" algorithm [14] provided by the data mining tool Weka 3 [15] to extract the words, which distinguish ARPs and programming posts. The "Information Gain Ratio" ranges between 0 and 1 and expresses the generative probability ranking of each word with respect to the type of post (ARP or programming post). In other words, it measures the ability of a word to split the population of posts correctly into the two main types of posts. Thus, a word with a higher "Information Gain Ratio" has a better ability to classify the posts between ARPs and programming posts because this word is more unique and common for one particular type of post. Section V presents a list of the top 20 words.

III. Architecture-Relevant StackOverflow Posts

In this section, we present the types of architecture-relevant posts on SO as the results to RQ1. We identified three main types of SO posts according to the concerns mentioned in the questions of a post. We named them *Pure Programming Posts, Architecture-Relevant Posts*, and *Cross Architecture/Programming Posts*.

Pure Programming Posts (PPPs) are posts with questions related to performing a programming activity. The answers to questions in PPPs often include source code fragments, step-by-step guidelines on how to code a feature, or lower level technology details. This type of post constitutes the majority of SO posts. Since these types of posts are not the focus of this paper, we do not discuss their details. The reader may refer to existing empirical studies on programming posts (e.g., [10]).

Architecture-Relevant Posts (ARPs) are posts with questions related to performing an architecture design activity. The questions in ARPs sometimes consider quality attributes and contextual factors, and the answers involve experience and knowledge about technology solutions, their differences and capabilities. This type of posts is the focus of this paper. Therefore, in order to better understand ARPs to support knowledge capturing and reuse, we classified ARPs based on two dimensions: (1) the purpose of the question, and (2) the solution type of the question. Along the *purpose dimension*, ARPs could be classified into the following sub-types:

1) *Solution Synthesis*: concerned with searching for suitable technology solutions, which have certain characteristics (such as technology features, quality attribute evaluation); address a design problem or context.
2) *Solution Evaluation*: concerned with assessing one or more proposed technology solutions. The evaluation of solutions could be done individually or through a comparison between different alternative solutions. In addition, several concepts are considered during evaluation, such as technology features, benefits and drawbacks, suitable use cases, and quality attributes.
3) *Multi-purpose*: this type of ARP comprise both types of posts, solution evaluation and synthesis. Several questions are asked within a single post.

Note that there might be other purposes for asking questions in ARPs. We identified the purposes based on our sample. On the *solution type dimension*, ARPs could be classified as follows:

1) *Technology Feature*: focus on specific features of a technology (such as deployment, authentication). For example, Gorton et al. [4] provide a taxonomy for features in database management systems.
2) *Technology Bundle*: consider the technology as a single architecture solution, without referring to the features within the technology. Technology bundle solutions are usually referred to using technology names (e.g., WCF).
3) *Architecture Configuration*: concerned with the components and connectors design configuration. The types of components and connectors could either belong to a technology feature (see above) or bundle (see above) or a conceptual solution (such as an architectural pattern or tactic). Note that conceptual solutions are not a category of ARPs above since we were focusing technology solutions. Also, we did not find many pure conceptual posts.
4) *Combined Solution*: concerned with different solutions types. The post may consider two or more of the aforementioned solution types.

Similar to the purpose dimension, there could be additional solution types. In this paper we listed types identified in our sample.

By combining the purpose and solution type dimensions, we can specify types of ARPs. For example, an ARP which is about evaluating a solution (purpose dimension) and discusses a technology bundle (solution type dimension) is a post, which is concerned with evaluating a technology bundle solution (e.g., a framework). On the other hand, an ARP about solution synthesis (purpose dimension) that discusses an architecture configuration (solution dimension) is concerned with searching for suitable component designs to a design problem at hand. Thus, the combination of both dimensions, three types in the purpose dimension and four types in the solution dimension, leads to 12 types of ARPs in total.

In order to clarify the types of ARPs within SO posts, we refined ARP's types into question variations. We present in tables II and III two examples for two variations in solution evaluation and synthesis ARPs. We quoted sentences from the ARP's questions and answers, and associated them with

TABLE II: Quotations from a *solution synthesis ARP*

Solution Synthesis ARP Question: • *Business Requirement* → "One of my clients has asked me developing a stock analysis program with close to 50 years of stock data for almost a thousand symbols" • *Design Issue* → "I've developed a series of filters (...). We want to run this filter for each day of data (...) We are figuring about 40 hours or so to run the report on our entire data (...) Does anyone have any general ideas or experiences with a web architecture that will support ultra-long asychronous processes?"
Solution Synthesis ARP Answer: • *Recommended ADD* → "I would recommend a standalone Windows Service (...) which would run constantly and check (poll) for 'jobs to process' in a database, then update the database with results and progress information." • *ADD Rationale* → "I've used it before many times and it's reliable, scalable and has good performance." • *Technology Drawback* → "keep web requests to a minute or two maximum - they were never designed for heavy processing times."

TABLE III: Quotations from a *solution evaluation ARP*

Solution Evaluation ARP Question: • *Design Issue* → "There a lot of different ways a Silverlight application can connect back to it server" • *ADD Alternatives* → "How do I choose between WCF, REST, POX and RIA services for a new Silverlight application"
Solution Evaluation ARP Answer: • *ADD Rule* → "WCF seems best suited when the service can be viewed as the business layer of the application, that is, when your service has "intelligent" operations" • *Technology Benefit* → ".NET RIA Services hides all this. It uses WCF under the covers, but that is completely hidden. You don't have to write asynchronous code" • *Recommended Technology ADD* → "I would strongly advise anyone to create a service layer using WCF Web APIs"

formerly identified *AK concepts* [3], [6], which they might refer to (e.g. *Design Issue, ADD*). The study of AK concepts in ARPs, and their distribution among the different types of ARPs is subject for future work. In addition, we provide additional examples for the rest of the question variations online[4].

Cross Architecture/Programming Posts (CAPPs) are posts with questions which could be either asked to perform a programming or an architecture activity. The answers to these questions provide information about technology solutions relevant for architecting but also for programming. Examples are posts which are about searching or selecting programming tools (e.g., unit testing frameworks, IDEs). These posts could be of interest to both the architect and the programmer: The architect is concerned with the impact of programming tools on the productivity and quality of the software or how tools integrate with frameworks and programming technologies (languages, libraries, etc.), while the programmer is interested in the programming features of tools. This type of posts are not analyzed further within the next section. Nevertheless, they are of interest to be addressed in a future work.

Summary: Based on our analysis described in Section II, we categorized 1,653 (65.8%) posts to be PPPs, 769 (30.6%) posts as ARPs, and 89 (3.5%) posts as CAPPs. Fig. 1 shows the distribution of the different types of ARPs within our sample. The results show that the majority of the ARP posts (91.4%) has one single purpose, either solution synthesis or evaluation. In addition, most of the ARPs (449 posts, 58.4%) use technology bundles as their solution type. When comparing the distribution of posts between the evaluation and synthesis posts, we can observe that the combined solutions post types within the synthesis posts are more than double of the combined solutions post types within the evaluation posts. On the other hand, technology bundle post types within the evaluation posts are nearly 30% more than the technology bundle post types within the synthesis posts.

[4]https://goo.gl/EhXfxg

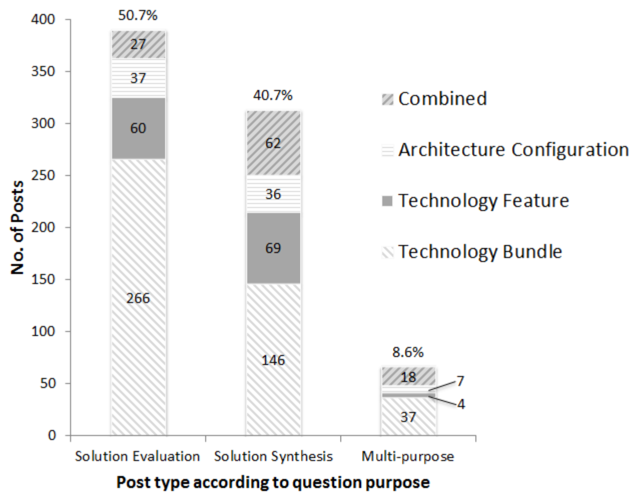

Fig. 1: Distribution of ARP types according to purpose and solution types of SO posts.

IV. Practitioner Evaluation

In this section, we present our result to RQ2. Participants classified posts into ARP or PPP, and also indicated their confidence. For each post, participants responded as follows:

1) *Yes - Architecture Post*: It is clear that the post is ARP.
2) *Maybe - Architecture Post*: The post is probably ARP.
3) *Do not know*: It is unclear if this is a PPP or ARP.
4) *Maybe - Programming Post*: The post is probably PPP.
5) *Yes - Programming Post*: It is clear that the post is PPP.

We calculated two measures to evaluate our classification:

1) *Agreement* between our own classification of posts into ARP and PPP and the classification of posts into ARP and PPP by practitioners (i.e., the percentage of posts for which researchers and practitioners agreed versus the total number of posts classified). We calculated in Table IV the agreement for each participant for ARPs and PPPs, and for all participants for each post type (Table IV - row "All"). In addition, we calculated the agreement across all posts (ARPs and PPPs) for each participant (Table IV - column "Total"). Moreover, we calculated the agreement for the different types of ARPs in Fig. 2 and 3.
2) *Confidence level of agreement*, which has been calculated for each post based on two levels of confidence ("Maybe" and "Yes"). We assigned "Maybe" a value of 0 and "Yes" a value of 1. We then summed up the number of posts with "Yes" for agreed posts for each participant. By comparing this sum to the total number of posts, we obtained the confidence in percent (e.g., if out of 20 ARPs 15 were classified with a confidence of "Yes" and 5 with "Maybe" then the overall confidence for the agreement of that participant on classifying ARPs was 75%). We calculated confidence for each participant and each type of post, and also for each post type across all participants (Table IV - row "All") and for each participant across all types (Table

TABLE IV: Agreement and confidence for each participant

Par. ID	Posts Given / Answered	PPPs Agreement & Confidence		ARPs Agreement & Confidence		Total Agreement & Confidence	
1	108/99	95.4%	95.5%	83.6%	92.7%	88.9%	93.9%
2	109/102	91.5%	85%	76.4%	87.3%	83.3%	86.3%
3	109/106	93.5%	85.1%	71.2%	87.3%	81%	86.3%
4	105/97	97.8%	91.3%	70.6%	59%	83.5%	74.2%
5	109/109	90.4%	86.5%	76.9%	75%	83.7%	80.8%
6	109/109	92.3%	86%	71.1%	69%	81.73%	77.9%
7	110/89	93.2%	50%	84.4%	24%	88.8%	37%
8	110/110	100%	96%	73.2%	89.3%	85.71%	92.4%
9	101/91	95.6%	82.2%	95.6%	78.3%	95.6%	80.2%
10	101/101	95.6%	78.3%	57.1%	51%	75.79%	64.2%
11	102/94	69.6%	84.8%	89.6%	85.4%	79.8%	85.1%
All	1173/1107	92.3%	83.6%	76.9%	71%	84.3%	77.1%

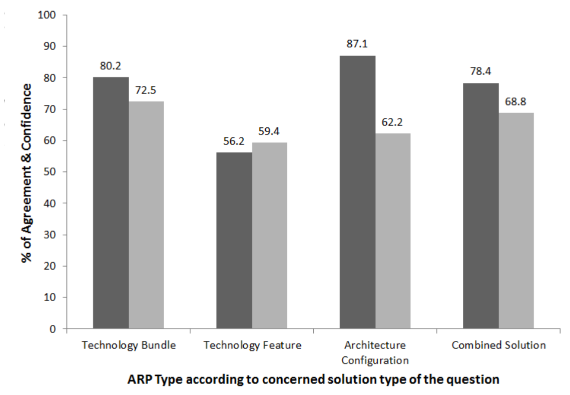

Fig. 2: Agreement and confidence for ARP types according to the solution type.

IV - column "Total") and for the different types of ARPs (Fig. 2 and 3).

Table IV shows the classification results of the different participants. Column "Posts Given/Answered" shows the number of posts given to the participants ("Given") and the number of classified posts ("Answered"), i.e., the "Given" posts excluding the posts, which the participants marked as "Do not know". The PPPs, ARPs, and Total Agreement and Confidence columns show the percentage of agreement and confidence across two groups dimensions, participant and post type.

We calculated the total agreement and confidence levels across the different ARP types, i.e., question purpose (evaluation, synthesis, and multi-purpose) and solution type (technology bundle, technology feature, architecture configuration, and combined), without differentiating participants. Fig. 2 and 3 illustrate the agreements and confidence levels.

We can observe that ARPs, which were classified by the authors to be evaluation or multi-purpose posts have high agreement and confidence, while the ARPs which were classified by the authors to be synthesis posts have a moderate agreement and confidence. On the other hand, ARPs which were classified by the authors to involve technology bundles have high agreement and confidence, while the ARPs, which were classified by the authors to involve technology features have a lower agreement and confidence. Moreover, architecture configuration posts have the highest agreement among solution types. However, they have moderate confidence level.

TABLE V: Top 20 distinctive terms between the ARPs and PPPs in our sample

Title term and gain ratio		Question term and gain ratio		Answer term and gain ratio	
soa	0.171	scalability	0.176	throughput	0.177
alternatives	0.169	compared	0.175	scaling	0.168
versus	0.167	pros/cons	0.162	xmpp	0.162
comparison	0.151	subscribers	0.162	cloud	0.159
lightweight	0.151	pros	0.159	scalable	0.155
notification	0.151	cons	0.159	tibco	0.148
choosing	0.151	meet	0.154	broker	0.141
distributed	0.148	real-world	0.151	esb	0.14
apps	0.148	amqp	0.151	enterprise	0.14
real-time	0.144	xmpp	0.148	governance	0.14
backend	0.144	mule	0.148	messaging	0.137
oriented	0.144	decision	0.124	brokers	0.136
pros	0.144	soa	0.116	redis	0.136
cons	0.144	corba	0.116	soa	0.132
ready	0.144	messaging	0.11	lightweight	0.129
middleware	0.144	scalable	0.106	scalability	0.128
share	0.14	balancing	0.106	udp	0.128
experience	0.14	availability	0.106	mule	0.128
dto	0.14	dtos	0.106	activemq	0.127
communicate	0.14	alternatives	0.104	bus	0.126

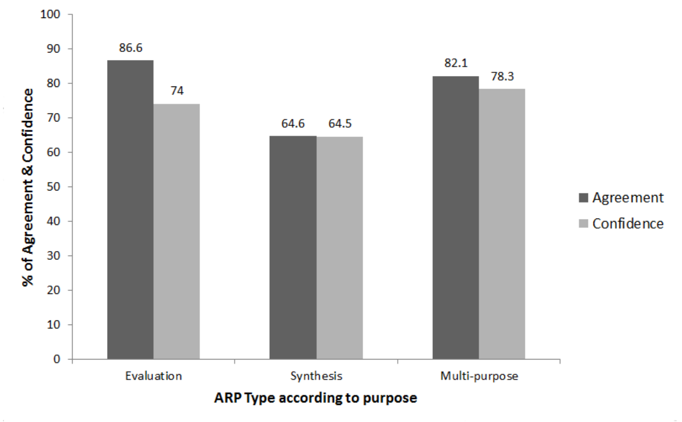

Fig. 3: Agreement and confidence for ARP types according to the purpose of the question.

V. Distinctive Terms between Architecture and Programming Posts

In this section, we present our result to RQ3. Based on the analysis process described in Section II, we present in table V the top 20 terms. The distinctive terms in the post title, question, and answers are represented as separate columns in table V, and the gain ratio is associated with each term. The terms are sorted in descending order according to their gain ratio.

From table V, we notice that many of the mentioned terms might refer to existing architectural concepts. For example, words such as "scalability", "throughput" and "availability" are amongst the top distinctive words, which are used usually to refer to quality attributes, while words such as "pros/cons", "compared" and "decision" might refer to a decision making situation, where the architect needs to decide between two or more architectural solutions. In addition, we notice that architecture patterns (e.g., "broker", "soa", "messaging") are part of the list of words to distinguish ARPs from programming posts. Note that the terms do not express a strict relationship between the terms and architectural concepts. This requires additional context analysis. However, the mentioned observations could give an indicator for the existence of such relationships.

VI. Discussion

A. Interpretation of Results

The results of our study indicate that it is possible to identify technology-related AK in SO. We found that most of the ARPs belong to one of two types according to the purpose of the question: solutions synthesis and evaluation. Both are common software architecture design activities [16]. This finding is a positive indicator for the suitability of capturing and reusing of AK for technology decisions from SO posts.

By analysing the agreement across ARPs, we find that ARPs related to evaluation have a higher agreement and confidence level than synthesis posts. One explanation could be that due to the richness of AK within the evaluation posts (e.g., benefits and drawbacks of solutions, decision rules). This type of knowledge embodies important factors for taking an ADD. On the other hand, the classification of some other types of posts differed among practitioners. For example, 56.2% of practitioners agree with our classification of technology feature posts for being ARPs. These results show that there could be other aspects within these types of posts which have influenced the classification decision of practitioners. The analysis and discovery of these aspects is subject to future work.

The list of distinctive ARP terms show to align with existing software architecture concerns and solutions (e.g., quality attributes, architectural patterns). This possible relationship provides additional validation for our classification between ARPs and PPPs, and indicates the possibility of analysing and capturing knowledge within ARPs.

B. Threats to Validity

1) Reliability: By the end of our categorization of ARPs, we made a final reliability test, and calculated Cohen's Kappa reliability coefficients [11] among 10% of our sample posts. The reliability coefficient for the classification between ARPs and PPPs is 0.9, while the classification of the ARPs into subtypes has a reliability coefficient of 0.75. This indicates reliability and agreement between classifications beyond chance.

2) Validity: The generalizability of our results could be a limitation, as our conditions for gathering our sample, as well as the sample size may not be representative for all possible ARPs in SO. However, the results could be an initial hypothesis for other future studies on ARPs. During our evaluation phase with practitioners, we did not select our participants randomly, but through our personal contacts. However, we considered their architecture experience as a factor for selection. In addition, the evaluation sampling method (stratified) of posts given to practitioners helped us cover all the types of posts, which made the evaluation more realistic. We collected our analysis posts based on keywords for technology names. Even though we considered the most popular technologies within the middleware domain, we cannot claim that our list technologies was exhaustive. There could be other posts related to less popular technologies, which were not included in our list of technologies. However, less popular technologies are most likely not discussed in many posts and therefore should not significantly impact our findings.

VII. Conclusion and Future Work

Our goal was to support existing AK management approaches with a more efficient method for capturing technology-related AK. We wanted to check if SO could be a viable source for reusable AK. Therefore, we conducted qualitative content analysis for a sample of SO posts. As a result of this analysis, we identified Architecture-Relevant Posts (ARPs) as one of the SO types. In addition, we further classified ARPs into several sub-types, which have been evaluated by practitioners. Our results showed that the identified ARP's types could be mapped to typical software architecture design activities. This finding does not only proof the usefulness of capturing AK from SO, rather it supports further analysis steps for the AK concepts within the SO posts. As a first step towards this direction, we performed additionally quantitative analysis for the distinctive terms between the classified ARPs and programming posts. The result list of distinctive words shows a possible relationship with existing AK concepts. In our future work, we will extend our analysis using additional analysis methods to further explore the different AK concepts and their relationships within ARPs, which will support building tools for automatically mining, classifying, and capturing AK from ARPs, in order to support AK management systems.

References

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

[2] A. Jansen and J. Bosch, "Software architecture as a set of architectural design decisions," in *WICSA*, 2005, pp. 109–120.

[3] 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.

[4] I. Gorton, J. Klein, and A. Nurgaliev, "Architecture knowledge for evaluating scalable databases," in *WICSA, IEEE/IFIP*, May 2015.

[5] F. Buschmann, K. Henney, and D. C. Schmidt, *Pattern-Oriented Software Architecture, Volume 4: A Pattern Language for Distributed Computing*. Chichester, UK: Wiley, 2007.

[6] M. Soliman, M. Riebisch, and U. Zdun, "Enriching architecture knowledge with technology design decisions," in *WICSA*, May 2015.

[7] C. Miesbauer and R. Weinreich, "Classification of design decisions: An expert survey in practice," ser. ECSA 2013. Springer-Verlag.

[8] G. von Krogh, "How does social software change knowledge management? toward a strategic research agenda," *The Journal of Strategic Information Systems*, vol. 21, pp. 154 – 164, 2012.

[9] D. Pagano and W. Maalej, "How do open source communities blog?" *Empirical Software Engineering*, vol. 18, no. 6, pp. 1090–1124, 2013.

[10] S. Nasehi, J. Sillito, F. Maurer, and C. Burns, "What makes a good code example?: A study of programming q & a in stackoverflow," in *ICSM*, Sept 2012, pp. 25–34.

[11] J. Cohen, "A Coefficient of Agreement for Nominal Scales," *Educational and Psychological Measurement*, vol. 20, no. 1, p. 37, 1960.

[12] P. Mayring, *Qualitative Content Analysis. Theoretical Foundation, Basic Procedures and Software Solution*. Beltz, 2014.

[13] R. Rosnow and R. Rosenthal, *Beginning Behavioral Research: A Conceptual Primer*. Pearson/Prentice Hall, 2008.

[14] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[15] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009.

[16] C. Hofmeister, P. Kruchten, R. L. Nord, H. Obbink, A. Ran, and P. America, "A general model of software architecture design derived from five industrial approaches," *Journal of Systems and Software*, vol. 80, no. 1, pp. 106–126, Jan 2007.