Analysis of Modern Release Engineering Topics

– A Large-Scale Study using StackOverflow –

Moses Openja
Dept. of Software Engineering
Polytechnique Montréal, Canada
openja.moses@polymtl.ca

Bram Adams
Dept. of Software Engineering
Polytechnique Montréal, Canada
bram.adams@polymtl.ca

Foutse Khomh
Dept. of Software Engineering
Polytechnique Montréal, Canada
foutse.khomh@polymtl.ca

***Abstract*—Release engineers are continuously required to deliver high-quality software products to the end-user. As a result, modern software companies are proposing new changes in their delivery process that adapt to new technologies such as continuous deployment and Infrastructure-as-Code. However, developers and release engineers still find these practices challenging, and resort to question and answer websites such as StackOverflow to find answers. This paper presents the results of our empirical study on release engineering questions in StackOverflow, to understand the modern release engineering topics of interest and their difficulty. Using topic modeling techniques, we find that (i) developers discuss on a broader range of 38 release engineering topics covering all the six phases of modern release engineering, (ii) the topics Merge Conflict, Branching & Remote Upstream are more popular, while topics Code review, Web deployment, MobileApp Debugging & Deployment, Continuous Deployment are less popular yet more complicated, (iii) - Particularly, the release engineering topic "security" is both popular and difficult according to data collected from StackOverflow.**

***Index Terms*—Modern Release Engineering, Topic Models, Empirical Study**

I. Introduction

Due to the market pressure, software industries are continuously required to deliver high-quality software products to the end-users faster. Unlike a few years ago, when software companies could work for months or even years on a release, many software companies now have only a limited time (e.g., a few weeks, days, or even hours) to ship their latest features to end users [1]. For instance, Google Chrome [2], Mozilla Firefox [3] and Facebook Mobile app have reduced their release "cycle time" to between two to six weeks, while Facebook web releases new features (1-2 times) a day [4].

Release engineering deals with all activities in between regular development and delivery of a software product to the end user. Through a series of phases such as code integration from the development branches, build & compilation, package, testing, and signing of the product for release, release engineers transform developers' source code into a product ready for users' consumption [5]–[7]. Releasing complex software systems with hundreds to thousands of users can be challenging and requires skills that are not always well mastered by developers and engineers. In fact, release engineers must implement continuous delivery and deployment practices and must be knowledgeable about specialised technologies and tools that support activities like continuous integration & source control management, testing, cloud provisioning, configuration management, application deployment, release orchestration [8]. It is therefore not surprising to see an increase of the prevalence of discussions about release engineering practices and tools on Q&A online developer forums, such as Stack Overflow. Stack Overflow is the most popular Q&A forums for software development. As of May 2020, it has recorded more than 19 million questions and 29 million answers. Developers turn to Stack Overflow to seek answers from their peers about issues that they face when using different software development technologies. For example, in the Stack Overflow post ID 26440324, a developer asked about "Automated build on Docker Hub" as follows: *(1) I have created an account on Bitbucket which is attached to a repository (no team, no group, just a user on a prepository). (2) In Docker Hub, I tried to link to Bitbucket via button + Add Repository / "Automated Build". (3) I get logged in alright, but it says "No repos available". That is strange as I can see the repository when logged into Bitbucket with this specific user. I have created this Bitbucket user for the sole purpose of being able to see that repository.*[1]. This question received an answer only after over half a year. Which may be a signal that the issue faced by this developer is either rare and–or difficult to resolve or not interesting enough to Stack Overflow users. An analysis of release engineering Q&A discussions on Stack Overflow can provide insights about the prevalence, popularity, and difficulty of various release engineering topics and guide the research community towards developing better techniques and tools to support release engineers. Therefore, in this paper, we conduct a large-scale empirical study of release engineering related posts on Stack Overflow and apply topic modeling [9] to understand the discussion topics of release engineers and identify the most important challenges that they face. In particular, we answer the following five research questions.

• **RQ1. Release Engineering Topics:** *What topics are discussed around Release Engineering?* The goal of this research question is to identify the main issues experienced by release engineers when producing software. By using the Latent Dirichlet allocation (LDA) [9] technique, we group and label

[1]https://stackoverflow.com/questions/26440324/docker-hub-automated-build-linked-to-bitbucket

StackOverflow posts into 38 topics, from a total of $260,023$ release engineering questions and answers posted in a period of 11 years; i.e., from 2008 to 2019.

• **RQ2. Popularity of Release Engineering Topics.** *What topics are popular among the release engineers?* The goal of this question is to identify the most prevalent release engineering topics on Stack Overflow. We analyze the popularity of release engineering topics using 3 well-known metrics, and find the most popular topics to be: *Merge Conflict, Branching & Remote Upstream, and Feature Expansion*. This result suggests that despite the many version control systems and source code management tools that exist, developers still struggle with merge conflicts and branching issues.

• **RQ3. Difficulty of Release Engineering Topics:** *What topics are more and less difficult to find answers?* The goal of this question is to identify the topics that may be challenging for the release engineers. Using a set of 2 well-known metrics, we find that topics *MobileApp Debug & Deployment, Continuous Deployment and Docker* are among the most difficult, suggesting that novel tools and techniques may be needed to support release engineering teams performing these activities.

• **RQ4. Correlation between the Popularity and Difficulty of Release Engineering Topics:** *How do topic popularity and difficulties correlate?* Through this question we want to understand if popular release engineering topics are difficult to address. Results show that the topic *Security* is both popular and difficult, topics *Merge Conflict, Branching & Remote Upstream, Feature Expansion* are among the most popular, while topics *MobileApp Debug & Deployment, Continuous Deployment, and Docker* are among the most difficult. Release engineers must prevent malwares from infecting their products to be released and patch vulnerabilities quickly before they can be exploited by malicious users. This result suggests that release engineers may be struggling with these critical activities. More research, training, and investments are required to better support security management activities.

• **RQ5. Types of Release Engineering Questions:** *What types of questions do release engineers ask in StackOverflow?* This research question aims to deepened our understanding of questions asked by release engineers. We aim to understand for example why they choose StackOverflow over other sources of information, such as official documentations. By grouping the questions asked by release engineers into $How?$, $Why?$ and $What?$ categories, we find that release engineers frequently ask about $how?$ to do things; often seeking clarifications and explanations (i.e., $what?$), and less frequently questioning (i.e., $why?$) certain aspects of release engineering practices, techniques, and tools. The high percentage of questions in the category $How?$ suggests that release engineers need support to create working solutions. Our dataset is available at [10]

The remainder of this paper is organised as follows. In Section II, we describe the steps followed in our analysis. In Section III, we discuss the results of our analysis. Section IV discusses related works. In Section V, we discuss the implications of our findings for the software research community, practitioners and educators. Finally, we conclude the paper in Section VII.

II. Methodology

This section summarizes the steps we took to analyze StackOverflow questions and answers, and answer our research questions *RQ1–RQ5*.

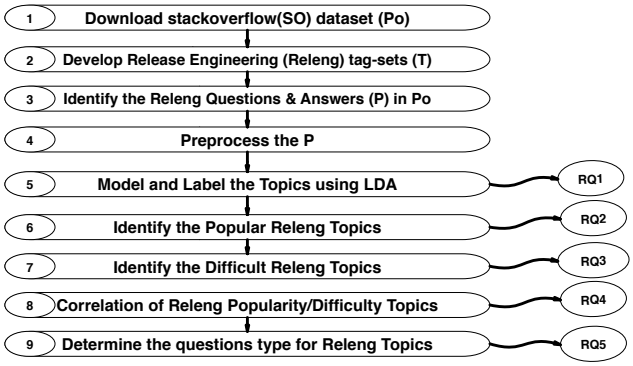

Fig. 1: Overview of our data analysis process.

• ***Step (1) - Download StackOverflow Dataset (P_0):*** In this first step, we used SOTorrent [11] to download the set of StackOverflow posts P_0, which contains both questions and answers and their metadata. The dataset P_0's metadata includes title, body, tags, creation date, views, identifier, question or answer type, title and favorite counts, score, and the identifier of the accepted answer of a post if the post is a question with accepted answers. A question can have at least one tag and a maximum of five tags. An answer to a question is an accepted answer if the developer who posted the question marked it as accepted.

The dataset P_0 is based on the official StackOverflow data dump released on 2019-09-04[2]. The dataset P_0 has a total 45,919,817 questions and answers for all the post made from September 2008 to September 2019 by $4,479,628$ developer participants of StackOverflow. Among these posts, $18,154,493$ and $27,765,324$ are questions and answers, respectively. In total, $9,531,288$ (34%) of these answers are marked as accepted answers.

• ***Step (2) - Develop Release Engineering (Releng) tag-set (T):*** To extract releng posts from the huge sets of posts using tags, we need a way of identifying the set of tags T used by developers when posting releng questions in StackOverflow. This tag set T is then used to identify and extract only the releng related Questions & Answers from our initial dataset P_0. To develop T, we defined the initial releng tag set T_i[3] based on modern release engineering process described in the previous work [6], [7] as follows:

(1) T_i include tags for the major phases of releng process, i.e., tags "Integration", "Continuous Integration", "Build System", "Infrastructure-as-Code", "Deployment" and "Release"

[2]https://archive.org/details/stackexchange

[3]https://drive.google.com/open?id=1upZfJ19d8cWGRJ-1Jbmbdg8ROSCsRMIu

[7]. Note that some tag naming was changed to match the naming in StackOverflow (e.g., Continous Integration changed to continuous-integration) after manually searching the tops posts tagged with the suggested tag name. **(2)** Identify and extract from the initial dataset P_0 all questions Q whose tags match those in T_i. **(3)** We extract the tags of questions in Q to construct a larger candidate tags set T_{i2}. **(4)** Lastly, we select only the tags t relevant to releng, using two sets of heuristics μ and ν, measuring respectively, the significance and relevance of a tag for a topic. These heuristics were also used in previous works [12], [13]. We removed tags that did not meet these criteria as follows.

$\langle significance \rangle$ $\mu = \frac{No.\ of\ questions\ with\ tag\ t\ in\ Q}{No.\ of\ questions\ with\ tag\ t\ in\ P_0}$

$\langle relevance \rangle$ $\nu = \frac{No.\ of\ questions\ with\ tag\ t\ in\ Q}{No.\ of\ questions\ in\ Q}$

We set thresholds limits and for every selected tag t, we considered the tag to be significantly relevant to releng if its μ and ν values are greater than or equal to our thresholds. To identify optimal threshold values for μ and ν, we iteratively experimented with values from intervals $0.05 - 0.35$ and $0.005 - 0.035$, respectively. During this experimentation, the authors assessed the quality of the extracted posts and discussed among them until reaching a consensus on the best candidate set. Finally, the optimal threshold values derived for μ and ν was used to select our final tag-sets and extract the studied posts. To illustrate further, we constructed from the initial tag-set T_{i2}: $\{T_{int}\}$, $\{T_{ci}\}$, $\{T_{bs}\}$, $\{T_{IaC}\}$, $\{T_{dep}\}$, $\{T_{rel}\}$ presenting all the six major phases of releng of integration, Continuous Integration, Build System, Infrastructure-as-Code, Deployment and Release, respectively. • T_{int} for example includes the major tags related to branching and merging of code changes between the developer branches and the main project branch, within a distributed version system.

We found that the optimal threshold values of $\mu = 0.3$ and $\nu = 0.010$ allow finding significantly relevant tags for tagset T_{int}, T_{ci} and T_{IaC}, while the threshold values of $\mu = 0.2$ and $\nu = 0.005$ allow finding significantly relevant tagsets for T_{bs}, T_{dep} and T_{rel}. The threshold values pairs of $\mu = 0.2$ and $\nu = 0.005$ are consistent with previous works [12]–[14]. Finally, we have a complete tagset T with a total of 64 tags, a union of selected T_{i2}, i.e, $T = T_{int} \cup T_{ci} \cup T_{bs} \cup T_{IaC} \cup T_{dep} \cup T_{rel}$. T is shown in Table I in light gray background. Note that some generic tags like *continuous-integration, build, deployment* also exist in our final tag sets T. Those are important to identify the possible releng posts that could have been left out, allowing us to cover a wide range. Also, some selected tags t may have the significant and relevant values that are within our threshold limit and yet was manually removed. The manual analysis was particularly done to discover tags that are much broader than just release engineering. Tags which were manually removed include among others *'environment-variables', 'transactions', 'merge', 'release'*[4]. Additionally, the use of tag set T does not prevent releng post with other tags that may not necessary be part of the set. For example a tag 'msbuild' may return a post with tag like 'build-tool'. The approach described in step 2 to develop the tag set T has been used by many other previous studies [13]–[16].

TABLE I: The Selected significantly relevant tag-sets (in light gray) for 6 releng phases.

$\langle \mu, \nu \rangle$	T_{int}: **Tags set for releng Integration**	# of tags
$\langle 0.3, 0.015 \rangle$	branching-and-merging, branching-strategy, merging-data, manifest-merging, git, branch, version-control, git-branch, feature-branch	9
$\langle 0.3, 0.010 \rangle$	branching-and-merging, branching-strategy, merging-data, manifest-merging, git, branch, version-control, git-branch, feature-branch, svn, merge, mercurial, git-merge, github	13
$\langle \mu, \nu \rangle$	T_{ci}: Tags set for releng Continuous Integration	# of tags
$\langle 0.3, 0.015 \rangle$	continuous-integration, continuous-delivery, azure-devops, continuous-deployment, gitlab, integration-testing, circleci, travis-ci, bitbucket, automated-tests, tfsbuild, jenkins	12
$\langle 0.3, 0.010 \rangle$	continuous-integration, continuous-delivery, azure-devops, continuous-deployment, bamboo, gitlab, integration-testing, circleci, travis-ci, tfvc, bitbucket, automated-tests, tfs2013, tfsbuild, jenkins	15
$\langle \mu, \nu \rangle$	T_{bs}: Tags set for releng Build System	# of tags
$\langle 0.2, 0.010 \rangle$	build,msbuild,build.gradle,phonegap-build,tfsbuild,build-process,build-automation	7
$\langle 0.2, 0.005 \rangle$	build, msbuild, interface-builder, build.gradle, flash-builder, phonegap-build, tfsbuild, build-process, scenebuilder, build-automation	10
(0.2,0.005)	build, msbuild,interface-builder, build.gradle, flash-builder, c++builder, phonegap-build, tfsbuild, build-process, query-builder, stringbuilder, scenebuilder, build-automation, processbuilder, powerbuilder	15
$\langle \mu, \nu \rangle$	T_{IaC}: Tags set for releng Infrastructure-as-code	# of tags
$\langle 0.3, 0.015 \rangle$	ansible, chef, puppet, ansible-playbook, chef-recipe, salt-stack, salt,ansible-inventory, cookbook, test-kitchen	9
$\langle 0.3, 0.010 \rangle$	ansible, chef,puppet, ansible-playbook, chef-recipe, salt-stack,salt, chef-solo, knife, ansible-inventory, cookbook, test-kitchen, terraform	13
$\langle \mu, \nu \rangle$	T_{dep}: Tags set for releng Deployment	# of tags
$\langle 0.2, 0.010 \rangle$	deployment,azure-devops,web-deployment,development-environment,environment,devops,production-environment	7
$\langle 0.2, 0.005 \rangle$	deployment, environment-variables, azure-devops, web-deployment, development-environment, environment, devops, production-environment, production, setup-deployment, azure-pipelines	11
$\langle \mu, \nu \rangle$	T_{rel}: tags set for releng Release	# of tags
$\langle 0.2, 0.010 \rangle$	release,rollback,maven-release-plugin,autorelease,release-management	5
$\langle 0.2, 0.005 \rangle$	rollback, maven-release-plugin, autorelease, release-management, nsautoreleasepool, ms-release-management, release-mode,azure-pipelines-release-pipeline,transactions	8

• ***Step (3) - Identify and Extract Releng Questions & Answers*** (P)***:*** Since StackOverflow is a general questions and answers website for engineering challenges, we now use the tag set T generated in step 2 to extract only the releng posts (questions and answers) P, to be used for the main goal of this paper. To do that, we consider a post as related to releng if its tag belongs to T. Applying this extraction criteria, we initially identified a total of $184,830$ questions and $196,299$ answers. From the extracted answers, $75,193(38.3\%)$ are marked as accepted. To construct P, we included only questions and the accepted answers, following the recommendation from other previous studies [12]–[14], [17]. This give us a final set with a total of $260,023$ questions and answers, where $184,830(71.0\%)$ are questions and the rest $75,193(29.0\%)$ are accepted answers.

• ***Step (4) - Preprocess Releng posts*** P***:*** This step, prepare our final posts P for the subsequent analysis of forming clusters of the posts. For every question, we first joined its title and body text to create one final body text. The extracted post text contains the HTML tags (for example, to present a paragraph, code snippet, URL, among others), so our first step was to clean those tags. We therefore remove all possible HTML tags which we could identify, such as $\langle p \rangle \langle / p \rangle$ and $\langle a \rangle \langle / a \rangle$, and code snippets surrounded by $\langle code \rangle \langle / code \rangle$. We then further clean the words identify as stopwords [18] such as numbers, 'a', 'the' and 'is', punctuation marks and non-alphabetical characters, as identified by MALLET's list-of-stopwords [5]. Finally, we used Porter stemmer [19] to reduce words to their stemmed representations. For example 'programmer' was reduced to 'program', 'configuration', 'configure' and

[4]https://drive.google.com/open?id=1s_ujZJZZ3YKuBAn4CWm_WR0tQsv10bxq

[5]https://github.com/mengjunxie/ae-lda/blob/master/misc/mallet-stopwords-en.txt

'configured' all were reduced to 'configr'.

• ***Step (5) - Model and Label Releng Topics:*** This step aims to extract the releng topics from the final set of prepossessed questions and answers P. To do that, we build topic models on P using the MALLET [20] toolkit implementing the Gibbs sampling algorithms [21] for latent Dirichlet allocation LDA [9]. LDA is a state of the art, widely adopted topic modeling technique, which models a topic as a set of frequently co-occurring words to approximate real-world situations [13]. Further, LDA is probabilistic. The posts are categorized into K topics after I iterations grouping. A topic is a vector of word probabilities, and a document is a vector of topic probabilities. A topic with the highest proportion value is the most dominant topic.

To improve the quality of the text during classification, we choose uni-gram and bi-grams following the previous studies that has shown that transformations such as bi-gram improve the quality [22]. Additionally, the number of topics K and the iterations grouping I are parameters set by user as a way to control the granularity of the discovered topics [17]. To help us discover the right number of topics K, we experimented with varying values of K ranging from 5 to 60 in increments of 5, and I varying from 500 to 3,000 with increments of 500. We aim to capture a broad range of topics within our dataset while keeping them distinct from each other. Our experiment with $K = 40$ topics and $I = 1,000$ returned meaningful releng topics of medium level of granularity. The topics number K and the iteration I settings is consistent with other previous studies [16], [17], [23] that used StackOverflow posts.

After generating these LDA topics grouping, the next steps were to manually assign a label to all the 40 returned topics groupings (i.e., the output of running MALLET). First, the documents were shared among the labeling team (the first and the second author of this paper). Each document in a group contains the corresponding probabilistic value scores and the top 30 keywords extracted from common, occurring words in the group of documents. Intuitively, the higher the probability value of a document, the higher its contribution to that topic's words, and hence its impact on assigning the topic label. To provide the right labels, we sort the documents by the dominant probability values from highest to lowest and randomly read through the top 15 to 20 documents representing each group. We used that information together with the top 30 keywords to assign a topic name to each group. During the topic labeling, the authors discussed and assigned labels after having a common agreement. The first author is a graduate student with many years of experience in software development, the second and third authors are both professors with extensive research experience in releng. Our final list of releng topics has a total of 38 topics and provides the answers to our **RQ1**. Table II shows the results of our topic label. The topic 9 and 21 were merged into one topic. Topics 38 and 39 were also merged because of their similarity.

• ***Step (6) - Identify Releng Popular Topics:*** This step aims to show the topics obtained in the previous steps that may be more popular among release engineers. To determine the popularity of a topic, we used three metrics adapted from previous works, as follows: average number of views [13], [15], [16], [24], [25], average total number of questions marked as favourite by users [13], [15], [24], [26], and the average question scores [13], [15], [24]–[26]. Intuitively, a more popular topic is the one that has the higher number of views, favorites, and a higher score. The results of step 6 is shown in Table III to answer **RQ2**.

• ***Step (7) - Identify Difficulty of Releng Topics:*** At this step we measure the difficulty of each topic to showcase how challenging they may be to release engineers and call for more attention. To measure the difficulty of a releng topic, we used two (2) known metrics adapted from previous works. The percentage of questions of a topic that have no accepted answers [12], [13], [15], [16], and the average of median time needed by the questions of a topic to receive an accepted answer [13], [15], [16]. Intuitively, a more difficult topic is the one having fewer accepted answers received after a long amount of waiting time. The analysis of Step 7 corresponds to our research question **RQ3**. Table IV shows the difficulty of releng topics.

• ***Step (8) - Determine the correlation of Popular and Difficult Releng Topics:*** This step identify the correlation between the difficult and the popular releng topics (if any). We used Kendall correlation tests to measure these correlations, since it is considered more stable (since it is less sensitive to outliers). We investigated 6 correlations between the 3 popularity metrics and the 2 difficulty metrics for the releng topics discussed in step 6 and step 7. The results of step 8 is shown in Table V and answers **RQ4**.

• ***Step (9) - Determine the Types of Questions in the Topics:*** To identify the types of questions asked by release engineers on StackOverflow, we used a qualitative coding on a statistically significant random sample of releng questions. We chose for every high-level topic's category shown in Figure 2, a random sample. We chose a sample size corresponding to a 95% confidence level and a 5% confidence interval. In total, our random sample has 2,646 questions distributed as: Integration, CI/CD, IaC, Build System, Deployment, release, general topics: = 381, 382, 378, 381, 378, 370, 376, respectively. During the coding process, the authors assigned a category to each question using the following labels : 'How?', 'Why?', and '$What$?'. The labels are defined as follows. **(1)** A How? type of question seeks better ways to achieve a result. These type of questions ask for guides on how something can be done or the ways to set up an environment. For example *"how to build and run a console application within the build and have it write output files to the output directory of another project?"*. **(2)** A Why? type of question examines the reason, or the cause, or the purpose for an occurrence. Developers may ask for reasons why their proposed solution failed to work or why they have an error. An example includes *"Mercury Quick Test Pro and Virtual machines: Works from one client machine but not another,... If I access this virtual machine*

using another machine (using Remote Desktop), the script starts fine, but stops halfway through...Has anyone had this problem before, or have any idea why the behaviour is different between the two machines?". **(3)** Finally, the $What?$ type of question aims to get the information related to something (e.g., a clarification).For example, *"Exactly what is integration testing - compared with unit?"*

During our coding, to get the full sense of what a selected post is about, we directly read through the post from StackOverflow using the ID of the post. We then repeatedly go through the posts line per line before assigning the label as either How, Why or $What$ defined above. For the posts where we identify more than one type of question such as $How?$ and $Why?$, we discuss further to identify the most dominant theme and use it. We follow the coding approach used by Treude et al. [27], whereby for their case, they defined the types into 10, i.e., *"how-to, discrepancy, environment, error, decision help, conceptual, review, non-functional, novice, and noise"*. In this case, they were interested in the nature of the question; unlike our case, where we are interested in the general concept. For example, a question which could be *decision help, conceptual, non-functional* fall under the $What?$ category in our study. Finally, we label the questions that are not identified under any of the above categories, as $others$. The results of step 9 is shown in Figure 5 and answers **RQ5**.

III. Results

A. *RQ1: Release Engineering Topics*

Table II shows the topic name, the category, and the top 10 words for releng topics that release engineers discuss in StackOverflow. As explained in step 5 of our methodology, topics $No.9$ is merged with $No.21$ into a single topic *Build Failure*, while topic $No.38$ is merged with $No.39$ into the topic *Merge Conflict*.

As one can see in Table II, there is a broad range of 38 releng topics that release engineers ask. These topics span across all phrases of releng. We present the topics both at a higher and lower level of granularity. For example, topics *Merge Conflict*, *Web Deployment*, *Web UI Testing*, *Ansible* are fine-grained whereas topics *Software Testing*, *Security* are coarse-grained. In Figure 2, we give a visual presentation of the releng topics. For each category and topic, we indicate the percentage of questions asked, arranging them from the highest in the left, to the lowest in the right. The percentage score for a topic indicates how much the topic dominates among others. Summing these scores gives a total percentage for the category. Figure 2 was constructed by repeatedly arranging similar topics into groups. We combined the two phases of 'release' and 'production' due to the fewer topics discovered relating to them.

We can see that, the most dominating topics by category are *"Continuous Integration/ Continuous Deployment (CI/CD) (*21.9%*)"*, followed by *"Build System (*19.9%*)"*, then *"Integration (*18.3%*)"*, *"General Topics/ Different Environment* (12.6%)*"*, *"Deployment (*11.1%*)"*, *"Infrastructure-as-Code*

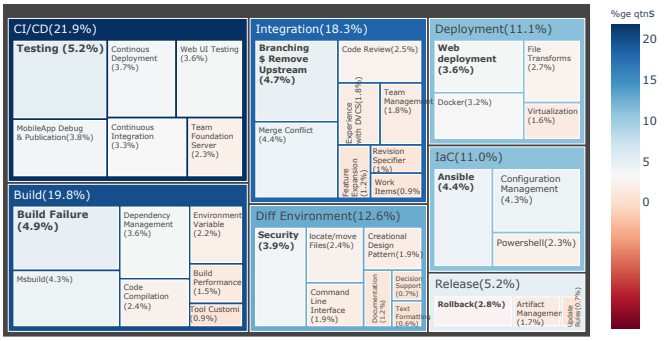

Fig. 2: Percentage of questions asked in Release Engineering Topics

(IaC) (11.0%)*"* and finally *"Release* (5.2%)*"*. This may suggest that many IT companies are adapting to CI/CD practice, yet they still find it challenging. Hence maybe a reason for its questions dominating among the rest. Figure 2 also shows that the topic *Software Testing* is dominating in CI/CD category, while *Build Failure* dominates in the Build System category. *Branching & Remote Upstream* dominates in the Integration category, *Security* dominates across the different environment, *Web Deployment* dominates in Deployment, and *Rollback* dominates in the Release category.

> ***Summary of findings (1)***:- Release Engineers ask about a broad range of 38 topics, where most of the questions are related to *Continuous Integration/Continuous Deployment (CI/CD) (*21.9%*)*. Only few questions ask about *Release (*5.2%*)* in StackOverflow.

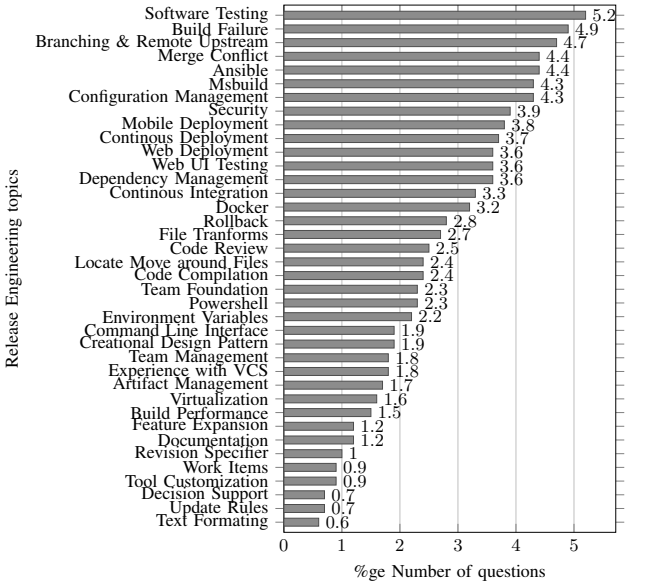

Fig. 3: Topics and percentage number of their questions.

To illustrate and give meaning to the topics above, we discuss them further by referring to their questions, for the topics identified as most dominating.

Software Testing: Figures 2 and 3 show that the topic *Software Testing* is the most dominating in the *CI/CD* category;

TABLE II: Release Engineering Topic Label, Category, and their top 10 stemmed words separated by a commas

No	Topic_Name	Category	Topic Words
0	*Build Performance*	Build	issue, time, problem, fix, start, happen, solve, stop, wait, long, day, close, bug, minute, hour, finish
1	*Environment Variables*	Build	set, environment, variable, configuration, default, property, setting, define, global, override, configure
2	*Team Foundation*	CI/CD	version, source, check, control, late, tfs, system, support, update, upgrade, tool, team_foundation
3	*Revision Specifier*	Integration	multiple, number, specific, single, time, revision, separate, date, give, mercurial, current, part
4	*Dependenc Management*	Build	project, dependency, library, include, main, jar, ant, share, maven, external
5	*Security*	Diff. Environment	user, access, key, private, account, public, password, permission, credential, login, username, security
6	*Web UI Testing*	Continuous Integration	page, open, click, show, select, link, view, element, display, browser, text, button, puppeteer, form
7	*Ansible*	IaC	task, ansible, list, host, module, playbook, template, group, variable, role, define, condition
8	*File Transforms*	Deployment	file, content, ignore, include, modify, exclude, xml, replace, rename, edit, filename
9*	*Build Error Debug*	Build	find, follow, type, search, give, resolve, information, miss, problem, documentation
10	*Web Deployment*	CD & Deployment	server, application, deploy, web, deployment, publish, site, azure, website, setup
11	*Virtualization*	Deployment	run, machine, window, start, setup, locally, slave, virtual, running, successfully, computer, runner
12	*Experience with VCS*	Integration	tag, svn, system, tool, large, time, subversion, lot, big, small, good, free, easy, people
13	*Locate Move around Files*	Diff. Environment	folder, copy, directory, path, delete, file, root, location, structure, move, workspace
14	*Team & Project Management*	Distributed Development	team, production, developer, development, code, dev, good, manage, live, product, work
15	*Feature Expansion*	Dev. Environment	change, make, apply, back, modify, switch, original, edit, state, modification, detect
16	*Decision Support around Releg.*	Asking for Guides	base, good, approach, common, easy, handle, component, provide, simple, require
17	*Software Testing*	Continuous Integration	test, integration, unit, testing, automate, write, case, selenium, continuous, result
18	*Docker*	CD and Deployment	service, image, docker, instance, server, container, connect, start, host, connection
19	*Code Review*	Distributed Development	request, pull, send, post, email, status, url, comment, receive, action, event
20	*Continuous Integration*	Continuous Integration	build, agent, vst, definition, building, teamcity, tfs, successful, bamboo, drop
21*	*Build Error*	Build	error, fail, follow, message, exception, throw, unable, failure, occur, log
22	*Documentation*	Diff. Environment	question, answer, understand, bit, read, point, thing, explain, difference, similar
23	*Continuous Deployment*	CI/CD	job, jenkin, pipeline, plugin, trigger, parameter, configure, pass, groovy, slave, schedule
24	*Msbuild*	Buld System	solution, studio, project, visual, msbuild, reference, target, assembly, framework, core, nuget, dll
25	*Rollback*	Release	transaction, rollback, follow, error, call, state, row, insert, work, execute, set, fail, run, record, minion
26	*Text Formatting*	Diff. Environment	case, order, level, note, end, match, rail, block, rule, top, part, simply, fact, space, side, pattern
27	*Code Compilation*	Build	code, generate, compile, report, include, tool, source, link, coverage, binary, static, result, compiler, cmake
28	*Tool Customization*	Build System	add, option, custom, remove, import, edit, enable, follow, extension, section
29	*Artifact Management*	Release	release, step, download, stage, artifact, debug, azure_devop, follow, mode
30	*PowerShell*	IaC	script, execute, process, run, write, program, command, powershell, execution, bash
31	*Work Items*	Distributed Team	work, fine, problem, item, thing, idea, make, correctly, perfectly, properly
32	*Creational Design Pattern*	Diff. Environment	call, method, return, function, object, load, code, pass, memory, array
33	*Update Rules*	Prod Environment	create, update, exist, automatically, check, manually, auto, empty, automatic, follow
34	*MobileApp Debug & Deployment*	Development and Release	android, app, problem, application, phonegap, device, find, support, platform, show
35	*Branching & Remote Upstream*	Branching & Merging	repository, push, local, remote, repo, clone, pull, fork, git, bitbucket, origin, fetch, submodule, locally,
36	*Command Line Interface*	CD, Build	command, line, output, follow, log, show, result, console, print, give
37	*Configuration Management*	IaC	package, instal, install, chef, client, resource, puppet, cookbook, attribute, installation
38**	*Merge Conflict*	Integration	branch, merge, master, feature, conflict, develop, trunk, rebase, back, delete
39**	*Git Revert Code Changes*	Integration	commit, history, point, git, tree, make, parent, back, current, index, patch, changeset, head,
			CI=Continuous Integration, CD=Continuous Delivery, IaC=Infrastructure-as-Code

representing 5.2% of questions that release engineers ask in StackOverflow. According to our dataset, the most viewed *Software Testing* question in StackOverflow is the question with identifier (Q_{id}: 5357601) *"What's the difference between unit tests and integration tests?[duplicate]... Are there different names for these tests? Like some people calling unit tests functional tests, etc?"*. This question is the most viewed with 222K, fourth most marked favourites and most scores (in our CI/CD category), and is a duplicate of question Q_{id}: 10752. *Software Testing* questions about "Spring MockMVC" like *"How to test a spring controller method by using MockMvc?"*, questions on "automation test for IOS" like Q_{id}: 20730917, 402389 "Setting up appium for iOS app test automation" may be due to poor documentation. While *Software Testing* is the most important contributor to quality assurance of the software product and must be performed at different stages of software development. Our study only gives a general view of the area that is challenging to release engineers, understanding the exact challenges in software testing call for future study. Other releng topics that may be interesting to investigate more in the future (in CI/CD category), according to our analysis are *"MobileApp Deployment", "Web UI Testing"*.

Build Failure:- Figures 2 and 3 shows that the topic *Build Failure* is the most dominating in the Build System category and the second most dominating releng question (4.9%) in StackOverflow. Examples of questions related to *Build Failure* include Q_{id}: 6383953 *"We have a situation where our builds have stopped executing in a stable manner. At a rate of about one every three we receive either TF215096 or TF215097 errors & the Build fails. If we then restart the Build controller, it works again - until next time...Server logs provide with little info, at least we 've found nothing that helps us resolve the situation. Various searches in the Net were also not productive..."*, Q_{id}: 41570435 *"Could not find com.android.tools.build:gradle:3.0.0-alpha1 in circle ci ... update the gradle plugin to the latest :... and this error occured.."*.Despite the abundant research that examined the causes of build failures, their automatic resolution, and proposed best practices [28]–[30], our results show that release engineers frequently seek help about *Build Failure*; which may suggest that they still struggle to obtain working solutions. Hence, more effort is needed to improve build systems, in particular, according to our findings shown on Figure 2 and Figure 3, more attention should be given to *"MsBuild", "Dependency Management", "Code compilation", "Environment Variables" and "Build Performance"*

Branching & Remote Upstream: Figures 2 and 3 show that *Branching & Remote Upstream* is the most dominating topic in the *Integration phase* category. It is the third most dominating topic (4.7%) that release engineers discuss on StackOverflow. This topic is about challenges in setting up repository branches and keeping them in sync. The most viewed question on this topic is (Q_{id}: 2765421) : *"...I want to be able to do the following: 1) Create a local branch based on some other (remote or local) branch...2) Push the local branch to the remote repository (publish), but make it trackable so git pull and git push will work immediately... I know about –set-upstream in Git 1.7, but that is a post-creation action. I want to find a way to make a similar change when pushing the branch to the remote repository."*. The question is the most viewed (3.4m) in

the topic *Branching & Remote Upstream*. It took an average of 9,480 hours to receive the right answers, and the second highest scores after question Q_{id}: 6591213 which is asking about changing branch name. Other examples of questions on this topic includes Q_{id}: 8345141 *"Keep histories in sync between local and remote mercurial repositories... The problem is then that the personal server-side clone is totally out of sync with the local repository because it was not rebased. We aren't using proper branches, so merge+rebase and transplant/graft seem to not be what we would use to get the repositories back in sync..."*.

Ansible:- is a lightweight general-purpose tool for automating the configuration of application deployment, cloud provisioning, and intra-service orchestration, among others [31]. Figures 2 and 3 show that the topic *Ansible* has the highest number of questions (4.3%) in IaC category and the fourth most number of releng questions in StackOverflow.

The most viewed questions about *Ansible* include Q_{id}: 32627624 *"Ansible: how to run task on other host inside one playbook? ... Is it possible to include one playbook in another? ..."*, and Q_{id}: 31152102 *"Is it a good idea to make Ansible and Rundeck work together, or using either one is enough? ... Seems both Ansible and Rundeck can be used to do configuration/management/deployment work, maybe in a different way So my questions are: ... If they can be used together, what's the best practice?... "*. The large number of questions about *Ansible* suggests that it may not be as easy to understand and use as reported in the literature [31]. According to our findings, other important topics related to the IaC category include *Configuration Management and Powershell*.

Security:- According to Figures 2 and 3, the topic *Security* is the most dominating topic in *Topic across different Environment* and eighth most dominating (3.9%) releng topic on StackOverflow. An example of question posted in StackOverflow about *Security* is Q_{id}: 56081795 *"how to restrict pipeline deployment by user?..right now, my pipeline works but anyone can trigger a deploy to any environment... the only thing I found is to block the entire project via user management, but it's not quite what I want to achieve... what I want to achieve is to have: User1 and User2 be able to deploy to production, but User3 can't trigger a Prod deployment, or at least group them and allow Group A to deploy and Group B to not deploy to Prod"*. There are other security questions related to 'automating the code signing and server certificate', i.e., Q_{id}: 22431526, 35821245, 23534429. Other general topics that cross between different environment in our findings, in order of dominance, include *Locate/ Move around Files, Command Line Interface, Creational Design Pattern, Documentation, Decision Support, and Text Formatting*.

Web Deployment:- Figures 2 and 3 show that *Web Deployment* is the most dominating topic in the Deployment category and the 11^{th} most discussed releng topic (3.6%) on StackOverflow. An example of *Web Deployment* question is Q_{id}: 49726235 which complain about a 'lack of tools': *"Web-based complex data-center automation tool..After evaluating existent tools like Ansible Tower, rundeck and others, it seems that no tool can fulfill the needed requirements. We have complex data-center servers, cluster of DB and web servers, the data-center has a lot of client-systems, +100, and other tools like solr, redis, kafka... deployed there across the physical servers, not to mention that the same data-center servers have different accounts, linux users, (QA,stag,production..etc), for now the meta-data about these environments alongside their web-apps, source code to be used, servers of the cluster are all defined on xml and there is a bash scriptsreads from that XML that operated manually to run any operation/task (like checkout the source, build, deploy, start, stop... and other customized operations)....And if this is not the right place, can you please point out where cat I ask such question?..."*. This question did not receive a right answer after two years and have not attracted many views. Other posts include Q_{id}: 51619 *"How to set up Git bare HTTP-available repository on IIS"*, Q_{id}: 47539115 *"How to GitLab Shared Runners deploy to server"*, Q_{id}: 33446277 *"Deploying the same site N times"*. "Troubleshooting Web Management Services" Q_{id}: 56559488, 56233303, 44929985, 54100188, 44573433, 37665540, 34457183, "Installing application in the remote server" Q_{id}: 33012702, 37221920, 54975830, "Web application and windows services" Q_{id}: 1789316, 9590422, 29808512 among others. Finally, other important topics in the *Deployment* category according to our findings include *"Docker", "File Transformation" and "Virtualization"*.

> ***Summary of findings (2)***:- *Software Testing, Build Failure, Branching & Remote Upstream, Merge Conflict, Web Deployment, Ansible & Configuration Management, Security and Rollback* are the top 9 releng topics discussed. They cover all releng phases. The *Software Testing* has the most asked questions and *Text Formatting* is least discussed topic on StackOverflow.

B. *RQ2: Popularity of Releng Topics*

Table III shows the results of our releng topic's popularity. The popularity of topics is measured as explained in step 6 of our analysis, and sorted by average total of views from the highest to the lowest. Intuitively, a more popular topic is one having more views, with more questions marked as favorites and receiving higher scores [12]–[15], [25]. According to Table III, the topic *Merge Conflict* has the highest average number of views, average number of favorite questions, and average number of scores. In contrast, topic *Web Deployment* has the lowest average number of views, sixth least favorites, and least scores. Also, topic *Creational Design Pattern* is the second least viewed, the least marked as favorite and the second least score.

> ***Summary of findings (3)***: - *"Merge Conflict", "Repositories Structure & Remote Upstream", "Feature Expansion"* of Integration category are among the most popular releng topics, while *Web Deployment* and *Creational Design Pattern* are amongst the least popular topics.

TABLE III: **releng Topics Popularity**

Topic_Name	Avg. Views	Avg. Favourites	Avg. Scores
Merge Conflict	9166.72	5.5	17.74
Branching & Remote Upstream	7618.43	3.6	10.83
Feature Expansion	6913.07	3.53	12.18
Security	5077.71	1.85	5.95
File Transforms	4988.98	3.00	10.03
Experience with VCS	4705.83	3.67	10.04
Locate Move around Files	4452.63	1.48	5.31
Team Foundation	4293.11	2.27	7.32
Tool Customization	4226.45	1.46	5.12
Update Rules	3957.25	1.87	6.32
Documentation	3892.68	2.41	6.23
Command Line Interface	3542.88	1.13	4.34
Web UI Testing	3365.99	1.28	4.78
Continuous Deployment	3066.3	0.71	2.9
Build Failure	3064.18	0.82	3.54
Environment Variables	3048.08	0.7	2.92
Text Formatting	3014.6	2.42	5.91
Msbuild	2937.23	1.05	4.36
Build Performance	2769.02	1.02	4.52
Configuration Management	2720.93	1.05	4.27
Code Review	2676.75	1.23	4.94
Dependenc Management	2649.33	0.95	3.42
PowerShell	2512.86	0.59	2.34
Revision Specifier	2452.26	1.25	4.58
Work Items	2444.42	1.00	3.61
Decision Support around Releg.	2377.9	1.67	5.22
Virtualization	2334.51	0.66	2.56
Rollback	2319.14	1.03	3.21
Artifact Management	2309.59	0.81	3.13
Docker	2254.28	0.87	2.94
Code Compilation	2222.25	2.84	3.39
Team & Project Management	2031.85	1.66	4.19
Ansible	2031.2	0.88	3.25
MobileApp Debug & Deployment	1923.66	1.01	3.32
Continuous Integration	1829.94	0.7	2.79
Software Testing	1749.85	0.93	2.99
Creational Design Pattern	1489.96	0.52	2.04
Web Deployment	1363.5	0.7	2.03
releng Topics	**3894.43**	**1.87**	**6.10**

TABLE IV: **releng Topics Difficulty**

Topic_Name	% No acc. Ansers	Hrs to acc. Answers
MobileApp Debug & Deployment	59.66	6.00
Continuous Deployment	59.09	9.00
Docker	59.09	6.00
Virtualization	58.58	9.00
Web Deployment	56.97	9.00
Build Failure	56.5	5.00
Code Compilation	55.78	9.00
PowerShell	54.76	3.00
Software Testing	54.2	7.00
Build Performance	53.74	4.00
Artifact Management	53.48	4.00
Code Review	53.35	4.00
Web UI Testing	52.42	4.00
Environment Variables	51.99	3.00
Command Line Interface	51.21	3.00
Dependenc Management	50.96	2.00
Update Rules	50.51	2.00
Security	50.44	4.00
Rollback	50.28	1.00
Configuration Management	49.61	4.50
Tool Customization	49.46	1.00
Decision Support around Releg.	48.92	2.00
Work Items	48.5	3.00
Locate Move around Files	47.84	1.00
Continuous Integration	47.74	11.00
Creational Design Pattern	47.32	0.00
Text Formatting	46.77	1.00
Ansible	46.33	6.00
File Transforms	45.52	1.00
Team & Project Management	43.9	1.00
Msbuild	43.39	8.00
Revision Specifier	43.01	3.00
Branching & Remote Upstream	41.61	0.00
Documentation	41.08	1.00
Feature Expansion	40.24	0.00
Experience with VCS	40.21	1.00
Team Foundation	39.73	1.00
Merge Conflict	39.12	0.00
releng Topics	**49.82**	**3.00**

C. ***RQ3**: Difficulty of Releng Topics*

Table IV shows the difficulties of the releng topics, which is measured as described in step 7 of our data analysis. Table IV is sorted in the order from the highest percentage number of questions without accepted answers.

"..., a topic with higher percentage of its questions not receiving accepted answers or taking longer time to receive accepted answers is more difficult" [14]. According to Table IV, topic *MobileApp Debug & Deployment* has the highest percentage of questions without accepted answers, and is in eighth position in terms of time required to receive an accepted answer. *Continuous Deployment* has the third-highest percentage number of questions with no accepted answers, and the third-highest average median time needed to receive an accepted answer. In contrast, topic *Merge Conflict* has the least percentage of questions with no accepted answers and the least amount of hours waited to receive accepted answers.

MobileApp Debug & Deployment:- Table IV shows *MobileApp Debug & Deployment* having the highest percentage of non-accepted answers (59.66% higher than the rest of the releng topics in StackOverflow). This may signal that it is difficult for release engineers to find answers related to *MobileApp Debug & Deployment* and therefore that it is a difficult topic. If we analyze question Q_{id}: 11934125 which is about "Automated testing of Hybrid apps" : *"If I want to perform automated testing of a PhoneGap app (for now, only on iOS), what options do I have (if any)?, I had also looked at using Jasmine with the Jasmine-jQuery plugin for much the same thing but it requires duplicating the app HTML (and the overhead of keeping the two in sync etc)"*. This question despite being popular did not receive a clear answer after more than seven years. The question was asked more than once (Q_{id}: 18739352) in StackOverflow. This situation may tell us (i) Mobile app developers are migrating from native apps to hybrid mobile development apps, due to its numerous benefits such as simplicity, portability, cross-platform support, reusing web development knowledge and cheap development processes [32], [33] among others. However, it seems that (2) they still lack efficient tools (e.g., for automation tests) to help in the migration and development of hybrid app.

Continuous Deployment:- Similarly, Table IV shows that *Continuous Deployment* related questions are difficult to answer, with the percentage of questions with no accepted answers being 59.09%. An example of *Continuous Deployment* question that took a very long time (4 years) to get an acceptable answer is Q_{id}: 31806108: *"Is there any method or plugins available to retrieve deleted Jenkins job? I have mistakenly deleted one job from Jenkins. So please give a suggestion to undo the delete"*. However, this difficult question was viewed over 20k times between August 2015 and January 2019.

> ***Summary of findings (4)***:- *"MobileApp Debug & Deployment" and "Continuous Deployment"* are among the most difficult releng topics.

D. ***RQ4**: Popularity/Difficulties releng Topics Correlation*

Our analysis indicates that topics such as *Merge Conflict*, *Branching & Remote Upstream* and *Feature Expansion* [34] are more popular in Table III, but the least difficult in Table IV. These results inspired us to investigate the correlations between popular and challenging topics. As explained in step 8 of our analysis approach, we investigate the relationship using

TABLE V: **Correlations of releng topics popularity/difficulty.**

coefficient / p-value	Avg. views	Avg. Favourites	Avg. Scores
% w/o acc. answer	**-0.338/0.002**	**-0.460/4e-05**	**-0.492/1e-05**
hrs to acc. answer	0.042/0.708	0.046/0.681	0.053/0.637

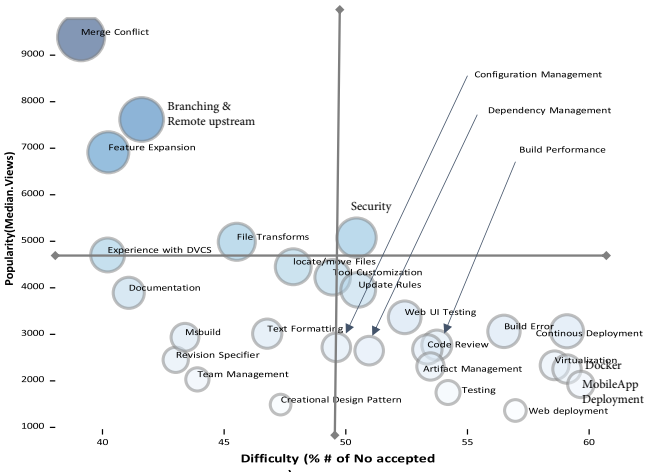

Fig. 4: Comparison of releng topics by popularity & difficulty

Kendall correlation by measuring the six correlations against the three popularity and two difficulty metrics of releng. Table V show the correlation measurement results of our analysis. The results in Table V shows a statistical negative correlation between the popularity and difficulty of releng topics even at 80% confident level. We further discuss the tradeoff between popularity and difficulty in Section V.

> ***Summary of findings (5)***:-There is a statistically significant negative correlation between the popularity and difficulty of releng topics.

Our results, however, do not imply that difficulty and popularity topics in StackOverflow are always negatively correlated. For example, Bagherzadeh and Khatchadourian [13] found that big data topics in StackOverflow are not negatively correlated. Similarly, the topic of general mobile development which is found to be popular by Barua et al. [17] is also seen as difficult by Rosen and Shihab [16].

*E. **RQ5**: Types of Releng questions*

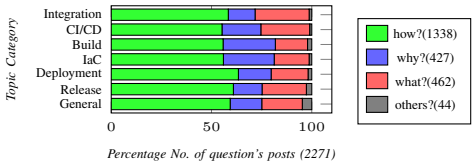

Fig. 5: Distributions of questions type in Releng topics category

Figure 5 shows the distributions of question types for the studied releng topic's category. We performed a Chi-squared test and found that there is no statistically significant difference between the types of questions across the topic's categories with a $p - value < 0.05$. Overall, as shown in Figure 5, the most prevalent type (58.9 %) of questions is "How", followed by "What" (20.3 %) and "Why" (18.8 %) for all releng categories. This result shows that release engineers are looking for more specific help to their problems, concepts, and errors. The high percentage of "$How?''$ questions in our results is consistent with the findings by Rosen and Shihab [16].

The majority of "$how?''$ questions are related to *'Deployment'* (63.42%). The category IaC and Build has more "why'' questions (26.84% and 25.33%, respectively), implying that better tools to help debugging could be most useful to improve releng pipelines, for example, Q_{id}: 32871956 asks "Why Aren't My Chef Normal Attributes Persisted?". The topic category *Integration* shows the most "$what''$ questions (26.58%); meaning that developers need some guidelines, documentation, and other useful information, to guide them in the integration phase of releng, for example, Q_{id}: 17466933 asks "What is the Git branching strategy with agile process?".

> ***Summary of findings (6)***:- Overall, the most prevalent type of questions is "How" (58.9 %), followed by "what" (20.3 %) and then "Why" (18.8 %), for all releng categories.

IV. Related Works

Generally speaking, release engineering remains one of the least studied areas in software engineering. Here we discuss some relevant release engineering works. We also discuss some works that used topic modeling. **(1) Release Engineering:-** Adams and McIntosh [7] summarized releng activities into six major phases, and outlined some research direction for the community. Wright and Perry [35] conducted semi-structured interviews with release engineers to understand why release process faults and failures occur and how companies recover from them, and how they can be foretold, and avoided. Castelluccio et al. [36] examined patch uplift operations in rapid release pipelines and formulate recommendations for improving their reliability. Khomh et al. [37] analyzed rapid release practices at Mozilla and found that despite the benefits of speeding up the delivery of new features to users, shorter release cycles can negatively impact software quality. **(2) Topic Modeling:-** Mehdi et al. [13] conduct a study to understand what big data developers discuss in StackOverflow. They used topic modeling techniques to identify 28 big data topics, and analyzed their popularity and difficulty. Anton et al. [17] used latent Dirichlet allocation (LDA) to analyze posts related to software development in StackOverflow. They compared their relationships and trends over time, to gain insights about the development community. Rosen and Shihab [16] used latent Dirichlet allocation (LDA) to summarized mobile-app related discussions in StackOverflow. They further determined the popular and challenging mobile-related issues, explored issues specific to the platform, and investigated the types of questions (e.g., what, how, or why) that mobile developers ask.

V. Implications and Discussion

So far, we have highlighted the different releng issues that are discussed in StackOverflow, pointed out the most popular and difficult topics, and finally, the types of questions that releng developers ask. In this section, we summarise and discuss the impactful issues and their implications for the researchers, practitioners, and educators.

Impactful Topics:- Although all the topics presented in this study are important in their specific area. We particularly summarized the releng popular and difficult topics, which we believe should be given more attention by researchers and practitioners. Figure 4 shows the popular topics in the y-axis as a measure of average views and the difficult topics in the x-axis as a percentage of questions with no accepted answers, and further divided into four quadrants showing more clearly, the relative popularity and difficulty of the issues. As can be seen in the top right quadrant, the topic *Security* is found to be both popular and difficult. This is therefore an important direction of research. Other topics we believe that need more attention as shown in Figure 4, include *Merge Conflict, MobileApp Deployment, Virtualization, Continuous Deployment, Debugging, Code Review, Web Deployment, File Transforms, Docker, Web UI Testing*. Identifying the actual problems related to each of these topics is out of the scope of this study and we call for more investigations by the research community. We also found that many developers are looking for the right solutions for the problems at hand (shown by the high number of "How" questions), concepts ("what" questions), and errors ("why" questions). This suggests the need for a general effort from both researchers, tool builders, or practitioner, and the educators.

To Researchers:- The empirical results of our study provide general views about the trends in releng; highlighting both the popular and challenging releng issues that are being discussed. We encourage researchers to tackle the top 10 difficult releng topics that we have identified, i.e., *MobileApp Debug & Deployment, Code Review, Docker, Continuous Deployment, Virtualization, Web deployment, Build Error Debug, Web UI Testing, Build System Performance, Security and Configuration Management*. According to Figure 4, the *Security* topic is both difficult and popular. This calls for more attention on releng security challenges. By infecting a build, malicious users could distribute their malware to thousand and even millions of users. Researchers should invest in developing efficient techniques and tools to support release engineers.

To Practitioners:- According to our findings, we recommend the team lead to always take the difficult topics in to consideration when they are distributing works between the project team members. Release activities related to *Security, Docker, Virtualization, Code Review and Continuous deployment* are more difficult and they should be assigned to the more experienced team members. Whereas the tasks related to code integration (with the exception of code review), could be assigned to even a less experienced team members.

To Educators:- Release engineering topics with higher percentage of questions on StackOverflow such as *"Software Testing"*, *"Branching & Remote Upstream"* and *"Merge Conflict"* should be better covered by trainings and course materials. Educators should also pay a particular attention to "Security" issues and ensure that challenges related to security in releng pipelines are covered in course materials.

VI. Threats to validity

To identify and extract the releng related posts in StackOverflow, we entirely relied on the selected releng tags, which is a threat. Our analysis may have missed to identify some releng posts. To mitigate this threat, we adapted methodologies used by many previous studies [13]–[16]. We are confident that this resulted in a significantly relevant releng tag set. Another threat is due to manual labeling, where we read the questions posts to appropriately map them to the topics. There is no tool that we could use to perform this task automatically. We further minimized this threat by both relating the topics keywords and randomly selecting at least 15 questions in the order of highest probability value to the category, and relating them to the user-defined tags. This approach has also been used by many previous works [13], [14], [24]. Another possible threat could be when choosing the optimal number of topics K and iterations value I. We, therefore, followed a well-defined technique adapted from [13], [16], [17], [24] together with multiple experiments to ensure that we identify reasonable K and I values. Also, LDA a probabilistic method may lead to random posts for the topics (to some extent), which is a threat. To mitigate this threat, we compared the returned 40 topics for the potential difference by running our final model at least four times; in all, we did not find a significant difference. Also, considering StackOverflow for understanding releng topics may be a threat. However, given the enormous popularity of Stack Overflow and the large number of release engineers using it, we believe that this threat is minimal. Nevertheless, future investigations with other crowed-source platforms are desirable to make our findings more generic.

VII. Conclusion

This paper presents the results of a large-scale study conducted using StackOverflow, to understand what release engineers ask about and identify the questions that are the most challenging during the six major phases of the release engineering process. We have identified popular and difficult topics and examined the relationship between them. We have categorized the questions that release engineers asked into three types (i.e., "$How?$", "$Why?''$, "$What?''$). These results can be extended to identify what is being asked for any topic.Finally, we have discussed our findings and formulated recommendations for researchers, practitioners, and educators teaching release engineering. Researchers can also apply our approach in a similar way to help them get data and information related to their study.

REFERENCES

[1] HP Application Handbook. (2012) Shorten release cycles by bringing developers to application lifecycle management. [Online]. Available: http://bit.ly/x5PdXl

[2] S. Shankland. (2010) Google ethos speeds up chrome release cycle. [Online]. Available: https://www.cnet.com/news/google-ethos-speeds-up-chrome-release-cycle/

[3] ——. (2011) Rapid-release firefox meets corporate backlash. [Online]. Available: http://cnet.co/ktBsUU

[4] C. 2014. (2015) Moving to mobile: The challenges of moving from web to mobile releases," keynote at releng 2014. [Online]. Available: https://www.youtube.com/watch?v=Nffzkkdq7GM

[5] S. McIntosh, B. Adams, T. H. Nguyen, Y. Kamei, and A. E. Hassan, "An empirical study of build maintenance effort," in *Proceedings of the 33rd International Conference on Software Engineering*, ser. ICSE '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 141–150. [Online]. Available: https://doi.org/10.1145/1985793.1985813

[6] B. Adams, S. Bellomo, C. Bird, T. Marshall-Keim, F. Khomh, and K. Moir, "The practice and future of release engineering: A roundtable with three release engineers," *IEEE Software*, vol. 32, no. 2, pp. 42–49, 2015.

[7] B. Adams and S. McIntosh, "Modern release engineering in a nutshell – why researchers should care," in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, vol. 5. IEEE, 2016, pp. 78–90.

[8] XebiaLabs. (2015) Periodic table of devops tools. [Online]. Available: https://xebialabs.com/periodic-table-of-devops-tools/

[9] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, no. null, p. 993–1022, Mar. 2003.

[10] M. Openja, "Release engineering posts," Aug. 2020, – A Large-Scale Study using StackOverflow –. [Online]. Available: https://doi.org/10.5281/zenodo.3980266

[11] S. Baltes, L. Dumani, C. Treude, and S. Diehl, "Sotorrent: Reconstructing and analyzing the evolution of stack overflow posts," in *Proceedings of the 15th International Conference on Mining Software Repositories*, ser. MSR '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 319–330. [Online]. Available: https://doi.org/10.1145/3196398.3196430

[12] C. Treude, O. Barzilay, and M.-A. Storey, "How do programmers ask and answer questions on the web? (nier track)," in *Proceedings of the 33rd International Conference on Software Engineering*, ser. ICSE '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 804–807. [Online]. Available: https://doi.org/10.1145/1985793.1985907

[13] M. Bagherzadeh and R. Khatchadourian, "Going big: A large-scale study on what big data developers ask," in *In Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE '19. City University of New York (CUNY): ACM, 2019. [Online]. Available: https://academicworks.cuny.edu/hc_pubs/524/

[14] S. Ahmed and M. Bagherzadeh, "What do concurrency developers ask about? a large-scale study using stack overflow," in *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3239235.3239524

[15] X.-L. Yang, D. Lo, X. Xia, Z. Wan, and J.-L. Sun, "What security questions do developers ask? a large-scale study of stack overflow posts," *Journal of Computer Science and Technology*, vol. 31, pp. 910–924, 09 2016.

[16] C. Rosen and E. Shihab, "What are mobile developers asking about? a large scale study using stack overflow," *Empirical Software Engineering*, pp. 1–32, 01 2015.

[17] A. Barua, S. Thomas, and A. Hassan, "What are developers talking about? an analysis of topics and trends in stack overflow," in *Empir Software Eng 19*. "-": Springer Link, 2014, p. 619–654. [Online]. Available: https://doi.org/10.1007/s10664-012-9231-y

[18] H. Joshi, J. Pareek, R. Patel, and K. Chauhan, "To stop or not to stop — experiments on stopword elimination for information retrieval of gujarati text documents," in *2012 Nirma University International Conference on Engineering (NUiCONE)*, 2012, pp. 1–4.

[19] K. Sparck Jones and P. Willett, Eds., *Readings in Information Retrieval*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997.

[20] A. K. McCallum, *A Machine Learning for Language Toolkit*, 2002. [Online]. Available: http://mallet.cs.umass.edu

[21] S. Geman and D. Geman, "Stochastic relaxation, gibbs distributions, and the bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, no. 6, pp. 721–741, 1984.

[22] C.-M. Tan, Y.-F. Wang, and C.-D. Lee, "The use of bigrams to enhance text categorization," *Inf Process Manag*, vol. 38, no. 4, pp. 529–546, 2002.

[23] A. Panichella, B. Dit, R. Oliveto, M. Di Penta, D. Poshyvanyk, and A. De Lucia, "How to effectively use topic models for software engineering tasks? an approach based on genetic algorithms," in *Proceedings of the 2013 International Conference on Software Engineering*, ser. ICSE '13. IEEE Press, 2013, p. 522–531.

[24] K. Bajaj, K. Pattabiraman, and A. Mesbah, "Mining questions asked by web developers," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, ser. MSR 2014. New York, NY, USA: Association for Computing Machinery, 2014, p. 112–121. [Online]. Available: https://doi.org/10.1145/2597073.2597083

[25] S. Nadi, S. Krüger, M. Mezini, and E. Bodden, "Jumping through hoops: Why do java developers struggle with cryptography apis?" in *Proceedings of the 38th International Conference on Software Engineering*, ser. ICSE '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 935–946. [Online]. Available: https://doi.org/10.1145/2884781.2884790

[26] G. Pinto, W. Torres, and F. Castor, "A study on the most popular questions about concurrent programming," in *Proceedings of the 6th Workshop on Evaluation and Usability of Programming Languages and Tools*, ser. PLATEAU 2015. New York, NY, USA: Association for Computing Machinery, 2015, p. 39–46. [Online]. Available: https://doi.org/10.1145/2846680.2846687

[27] C. Treude, O. Barzilay, and M. Storey, "How do programmers ask and answer questions on the web?: Nier track," in *2011 33rd International Conference on Software Engineering (ICSE)*, 2011, pp. 804–807.

[28] S. Mcintosh, M. Nagappan, B. Adams, A. Mockus, and A. E. Hassan, "A large-scale empirical study of the relationship between build technology and build maintenance," *Empirical Softw. Engg.*, vol. 20, no. 6, p. 1587–1633, Dec. 2015. [Online]. Available: https://doi.org/10.1007/s10664-014-9324-x

[29] F. Hassan, "Tackling build failures in continuous integration," in *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Nov 2019, pp. 1242–1245.

[30] N. Kerzazi, F. Khomh, and B. Adams, "Why do automated builds break? an empirical study," in *2014 IEEE International Conference on Software Maintenance and Evolution*, 2014, pp. 41–50.

[31] RedHat Ansible. (2020) Ansiblefest 2020 virtual experience. [Online]. Available: https://www.ansible.com/

[32] Native, "Web or hybrid mobile-app development. white paper," in *IBM Corporation. Document Number: WSW14182USEN*, no. WSW14182USEN, April 2012.

[33] I. Malavolta, "Beyond native apps: Web technologies to the rescue! (keynote)," in *Proceedings of the 1st International Workshop on Mobile Development*, ser. Mobile! 2016. New York, NY, USA: Association for Computing Machinery, 2016, p. 1–2. [Online]. Available: https://doi.org/10.1145/3001854.3001863

[34] J. Businge, M. Openja, S. Nadi, E. Bainomugisha, and T. Berger, "Clone-based variability management in the android ecosystem," in *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2018, pp. 625–634.

[35] H. K. Wright and D. E. Perry, "Release engineering practices and pitfalls," in *2012 34th International Conference on Software Engineering (ICSE)*, 2012, pp. 1281–1284.

[36] M. Castelluccio, L. An, and F. Khomh, "An empirical study of patch uplift in rapid release development pipelines," *Empirical Software Engineering*, vol. 24, no. 5, pp. 3008–3044, 2019. [Online]. Available: https://doi.org/10.1007/s10664-018-9665-y

[37] F. Khomh, B. Adams, T. Dhaliwal, and Y. Zou, "Understanding the impact of rapid releases on software quality - the case of firefox," *Empirical Software Engineering*, vol. 20, no. 2, pp. 336–373, 2015. [Online]. Available: https://doi.org/10.1007/s10664-014-9308-x