



Developer discussion topics on the adoption and barriers of low code software development platforms

Md Abdullah Al Alamin¹ · Gias Uddin¹ · Sanjay Malakar² · Sadia Afroz² · Tameem Haider² · Anindya Iqbal²

Accepted: 21 September 2022

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract

Low-code software development (LCSD) is an emerging approach to democratize application development for software practitioners from diverse backgrounds. LCSD platforms promote rapid application development with a drag-and-drop interface and minimal programming by hand. As it is a relatively new paradigm, it is vital to study developers' difficulties when adopting LCSD platforms. Software engineers frequently use the online developer forum Stack Overflow (SO) to seek assistance with technical issues. We observe a growing body of LCSD-related posts in SO. This paper presents an empirical study of around 33K SO posts (questions + accepted answers) containing discussions of 38 popular LCSD platforms. We use Topic Modeling to determine the topics discussed in those posts. Additionally, we examine how these topics are spread across the various phases of the agile software development life cycle (SDLC) and which part of LCSD is the most popular and challenging. Our study offers several interesting findings. First, we find 40 LCSD topics that we group into five categories: Application Customization, Database and File Management, Platform Adoption, Platform Maintenance, and Third-party API Integration. Second, while the Application Customization (30%) and Data Storage (25%) topic categories are the most common, inquiries relating to several other categories (e.g., the Platform Adoption topic category) have gained considerable attention in recent years. Third, all topic categories are evolving rapidly, especially during the Covid-19 pandemic. Fourth, the How-type questions are prevalent in all topics, but the What-type and Why-type (i.e., detail information

Communicated by: Kelly Blincoe, Mei Nagappan

This article belongs to the Topical Collection: *Mining Software Repositories (MSR)*

✉ Md Abdullah Al Alamin
mdabdullahal.alamin@ucalgary.ca

Gias Uddin
gias.uddin@ucalgary.ca

¹ DISA Lab, University of Calgary, Calgary, Canada

² Bangladesh University of Engineering & Technology, Dhaka, Bangladesh

for clarification) questions are more prevalent in the Platform Adoption and Platform Maintenance category. Fifth, LCSD practitioners find topics related to Platform Query the most popular, while topics related to Message Queue and Library Dependency Management as the most difficult to get accepted answers to. Sixth, the Why-type and What-type questions and Agile Maintenance and Deployment phase are the most challenging among practitioners. The findings of this study have implications for all three LCSD stakeholders: LCSD platform vendors, LCSD developers/practitioners, Researchers, and Educators. Researchers and LCSD platform vendors can collaborate to improve different aspects of LCSD, such as better tutorial-based documentation, testing, and DevOps support.

Keywords Low-code software development · Empirical study · Stack overflow

1 Introduction

There is a massive shortage of skilled software developers in this age of digitalization. According to Gartner, the demand for IT professionals will be multiple times more than supply (Waszkowski 2019; Torres 2018). To make matters worse, training and hiring new software developers are very expensive in this rapidly evolving world. LCSD aims to address this issue by democratizing software development to domain experts and accelerating the development and deployment process. It tries to bridge the gap between the system requirement and the developer constraints, which is a common reason for long development times in complex business applications.

LCSD is a novel paradigm for developing software applications with minimal hand-coding through visual programming, a graphical user interface, and model-driven design. LCSD embodies End User Software Programming (Pane and Myers 2006) by democratizing application development to software practitioners from diverse backgrounds (Di Sipio et al. 2020). It combines various approaches such as visual modeling, rapid app development, model-driven development, cloud computing, and automatic code generation. Low-code development tools enable the development of production-ready apps with less coding by facilitating automatic code generation. Additionally, LCSD platforms also provide more flexibility and agility, faster development time that allows responding quickly to market needs, less bug fixing, less deployment effort, and easier maintenance (Sahay et al. 2020; Di Sipio et al. 2020). These platforms are used to develop high-performance database-driven mobile and online applications for various purposes. As a result, low-code development is rapidly growing in popularity. According to Forrester, the LCSD platform market is estimated to reach \$21 billion by 2022. By 2024, over 65% of big companies will utilize LCSD systems to some extent, according to a Gartner report (Wong et al. 2019).

To date, there are more than 400 LCSD platforms (2022), offered by almost all major companies like Google (2020) and Salesforce (2022). Naturally, LCSD has some unique challenges (Sahay et al. 2020). Wrong choice of LCSD application/platforms may cause a waste of time and resources. There is also concern about the security/scalability of LCSD applications (Khorram et al. 2020). With interests in LCSD growing, we observe discussions about LCSD platforms are becoming prevalent in online developer forums like Stack Overflow (SO). SO is a large online technical Q&A site with around 120 million posts and 12 million registered users (Overflow 2020). Several research has been conducted to analyze SO posts (e.g., IoT (Uddin et al. 2021b), big data (Bagherzadeh and Khatchadourian 2019),

blockchain (Wan et al. 2019) concurrency (Ahmed and Bagherzadeh 2018), microservices (Bandeira et al. 2019)). The studies, however, did not analyze discussions about LCSD platforms in SO.

In 2021, we conducted an empirical study (Alamin et al. 2021) by analyzing 4,785 posts (3,597 questions + 1,118 accepted answers) from SO that contained discussion about nine LCSD platforms. The study offered, for the first time, an overview of the challenges software developers face while using LCSD platforms. However, to date, there are over 400 LCSD platforms and we observed discussions about many of those platforms in SO. Therefore, it was important that we revisit our empirical study with a larger dataset of discussions about LCSD platforms in SO. In addition, given that the previous empirical study was a conference paper, the analysis was not as in-depth as we could have provided due to space limitations. Therefore, a larger-scale empirical study of the challenges developers face to adopt and use the LCSD platforms was warranted. Such insights can complement our previous empirical study (Alamin et al. 2021) as well as the existing LCSD literature – which so far has mainly used surveys or controlled studies to understand the needs of low-code practitioners (Fryling 2019; Kourouklidis et al. 2020; Alonso et al. 2020; Khorram et al. 2020).

Specifically, in this paper, we present an empirical study of 33.7K SO posts relating to the top 38 LCSD platforms (according to Gartner (2022)) at the time of our analysis to ascertain the interest and challenges of LCSD practitioners. We answer five research questions by analyzing the dataset.

RQ1. What topics do LCSD practitioners discuss? Given that LCSD is a novel paradigm, it is vital to study the types of topics discussed by LCSD practitioners on a technical Q&A platform such as SO. As a result, we use the topic modelling method LDA (Blei et al. 2003) on our 33.7K post dataset. We find a total of 40 LCSD topics grouped into five categories: Application Customization (30% Questions, 11 Topics), Data Storage (25% Questions, 9 Topics), Platform Adoption (20% Questions, 9 Topics), Platform Maintenance (14% Questions, 6 Topics), and Third-Party Integration (12% Questions, 5 Topics). Around 34% of questions are particular to the many supported capabilities of LCSD platforms, while the remaining 66% are regarding development activities, namely application customization. This is because the LCSD platform's features are naturally oriented around a graphical user interface (GUI) in a drag-and-drop environment. As a result, any customization of such features that are not native to the LCSD platforms becomes difficult.

RQ2. How do the LCSD topics evolve over time? We elaborate on our findings from RQ1 by examining how the observed LCSD topics evolved in SO over time. We conduct an in-depth analysis of LCSD-related discussions from 2008 to mid-2021 in SO. We discover that since 2012, discussion about LCSD has piqued community interest, which has increased significantly throughout the pandemic, i.e., since 2020. In recent years, Platform Adoption-related discussions have acquired more traction than original application customization or database query-related discussions. Future research and LCSD platform vendors should support emerging topics such as Library Dependency Management, External Web Request Processing, Platform Infrastructure API, and Data Migration.

RQ3. What types of questions are asked across the observed topic categories? From RQ1, we find some of the unique challenges for LCSD practitioners regarding Customization, Data Storage on the completely managed cloud platforms. This motivates us to explore further to understand more of those challenges. For instance, we want to

understand if practitioners mostly ask about different solution approaches (i.e., How-type) or further explanation clarification type (Why/What-type). Following previous studies(Uddin et al. 2021b; Abdellatif et al. 2020), we manually annotated a statistically significant number of posts (e.g., 471 Questions) into four categories. We find that How-type (57%) is the most common form of inquiry across all five topic categories, followed by What-type (18%), Why-type (14%), and Other-type (12%) questions. Most of the How-type questions are application implementation-related, and most of the What-type and Why-type Questions are server configuration and troubleshooting related. According to our findings, proper documentation and tutorials might significantly reduce these challenges.

RQ4. How are the observed topic categories discussed across SDLC phases? Our findings from the previous research questions examined the practitioners' challenges on LCSD platforms and their evolution. The acceptance of this emerging technology depends largely on effective adoption into the various stages of a software development life cycle (SDLC). So, following our previous study (Alamin et al. 2021) we manually annotate statistically significant samples (e.g., 471 Questions) into six agile SDLC stages. We find that the Implementation (65%) is the most prominent phase in terms of the number of questions, followed by Application Design (17%) and Requirement Analysis & Planning (9.1%).

RQ5.What LCSD topics are the most difficult to answer? LCSD practitioners face many different challenges to understand different features of the cloud platform, server configuration. LCSD vendors aim to provide support from requirement gathering to deployment and maintenance, but practitioners still struggle with customization, data management, and cloud configuration. We find that, while the topic of application customization and the Implementation-SDLC are the most prevalent, Platform Adoption topic category and the Deployment-SDLC and Maintenance-SDLC as the most popular and hardest to get accepted answers.

This paper extends our previous paper (Alamin et al. 2021) along two major dimensions: the data used and the results reported. We offer details about the extensions below.

1. **Data (see Section 3).** The dataset in this study is significantly larger and more diverse than our previous paper as follows.
 - **Size.** The size of the SO dataset in this paper is almost seven times bigger than the dataset used in our previous paper. This study analyses 33766 posts (26763 Questions + 11355 Accepted Answers). Our prior paper examined 4,785 posts (3597 Q + 1188 A).
 - **Time.** This study analyzesLCSD-related discussions in SO between July 2008 to May 2021, while the previous study analyzed the discussions between July 2008 to May 2020.
 - **LCSD Platforms.** This study analyzes 64 LCSD-related tags which contain 38 LCSD platforms, while the previous study analyzed 19 SO tags related to 9 LCSD platforms.
2. **Empirical Study (see Section 4).** This paper considerably enhances our understanding of LCSD platforms over our previous paper (Alamin et al. 2021) as follows.
 - **Research Questions (RQ).** We have answered five research questions (RQ2, RQ3) in this paper compared to three RQs in our previous paper (RQ1, RQ4, RQ5). The two new RQs offer insights on the type of LCSD questions asked and the evolution

of the LCSD topics. Our revision of the previous three RQs provided several new results as follows.

- **LCSD Topics.** In this study, we found 40 topics organized into five high-level categories. We found 13 topics organized into four high-level categories in our previous paper. While we found all the previous 13 topics, we also found 27 new LCSD topics. This study found Platform Maintenance as a new high-level topic category (see Section 4.1).
 - **Finer-Grained Analysis.** Due to our use of more data, we find better results from our topic modeling. For example, some topics from our previous studies are broken down into more informative/coherent topics. For example, Client-Server Communication and IO from Platform Adoption topic category became topics Web Service Communication and Message Queue under Asynchronous Service to Service Communication sub-sub-category in this study as those topics contained more coherent discussions. Similarly, we have expanded our understanding of software development lifecycle phases (SDLC) around the new 40 topics (see Section 4.4).
 - **Topic Evolution.** Our new RQ2 analyzes the evolution of the observed LCSD topics in Section 4.2. We further discuss the prevalence and evolution of the topics across the top 10 LCSD platforms in our dataset (see Section 5.4).
 - **Question Type.** Our new RQ3 offers insights into the type of questions asked across the observed LCSD topics (see Section 4.3).
 - **Popularity vs Difficulty.** In addition to analyzing the popularity and difficulty of all 40 topics in Section 4.5, we also offer the following new insights. (a) Following a recent study (Uddin et al. 2021b), we report the popularity and the difficulty using two fused metrics (see Section 4.5). (b) We report the popularity and difficulty of the LCSD question types and SDLC phases (see Section 5.6)
3. **Related Work.** We have expanded our literature review with a comparison of key metrics around our observed LCSD topics against those previously reported for other domains while using the SO data (see Section 7.2).

Our study findings can enhance our understanding of the developers' struggle while using LCSD platforms. The findings would help the research community and platform vendors better focus on the specific LCSD areas. The practitioners can prepare for difficult areas. LCSD platforms can design more effective and usable tools. All stakeholders can collaborate to provide enhanced documentation assistance. The LCSD vendors can support increased customization of the LCSD middleware and UI to make the provided functionalities more usable.

Replication Package: The code and the datasets generated during this study is available in our public GitHub repository https://github.com/al-alamin/LCSD_challenge_EMSE

2 Background

This section aims to provide a high-level overview of LCSD development, as well as some of the relevant technologies and research that have shaped this industry. We hope that this will serve as a resource for future researchers (particularly those interested in the underlying technologies)) and practitioners to learn and contribute more to this emerging new field.

Low-code Software Application To cater to the demand of the competitive market, business organizations often need to quickly develop and deliver customer-facing applications.

LCSD platform allows the quick translation of the business requirement into a usable software application. It also enables citizen developers of varying levels of software development experience to develop applications using visual tools to design the user interface in a drag-and-drop manner and deploy them easily (Low-Code Development Platform 2022). LCSD is inspired by the model-driven software principle where abstract representations of the knowledge and activities drive the development, rather than focusing on algorithmic computation (Sahay et al. 2020). LCSD platforms aim to abstract away the complexity of testing, deployment, and maintenance that we observe in traditional software development. Some of the most popular low-code platforms are Appian (2022), Google App Maker (2020), Microsoft Powerapps (2022), and Salesforce Lightning (2022).

Technologies that Shaped LCSD Model-driven Software Engineering (MDSE) field proposes the adoption of domain-specific modeling practices (Brambilla et al. 2017). Low-code platforms adopt model-driven engineering (MDE) principles as their core that has been applied in several engineering disciplines for the purpose of automation, analysis, user interface design (Botterweck 2006; Pleuss et al. 2013; Brambilla et al. 2017) and abstraction possibilities enabled by the adoption of modelling and meta modeling (Basciani et al. 2014). Besides, End-User Development (EUD) is a set of methods, techniques, and tools that allow users of software systems, who are mainly non-professional software developers, at some point to create, modify or extend a software artifact (Paternò 2013; Fischer et al. 2004). EUD for GUIs can be a good example of its usage (Costabile et al. 2007). Scratch (Resnick et al. 2009), Bloqqi (Fors 2016), EUD-MARS (Akiki et al. 2020), App Inventor (Wolber 2011), AppSheet (AppSheet 2021) are such “low-code/no-code” application development tools that offer visual drag-and-drop facilities. Similarly, there are several other research areas within the domains of HCI (Sinha et al. 2010) and Software engineering, such as Visual Programming (Burnett and McIntyre 1995), Programming by example (Halbert 1984), End users programming (Myers et al. 2006), domain specific language (Mernik et al. 2005; Van Deursen et al. 2000), trigger action programming (Ur et al. 2014) that aim to enhance the technologies underlying low-code software development. Thus, gaining a better knowledge of the problems associated with low-code platforms through developer discussion would be extremely beneficial for further improving these studies.

Development Phases of an LCSD Application A typical LCSD application can be built in two ways (Sahay et al. 2020): 1. “UI to Data Design”, where developers create UI and then connect the UI to necessary data sources, or 2. “Data to UI” where the design of the data model is followed by the design of the user interfaces. In both approaches, application logic is implemented, and then third-party services and APIs are integrated. APIs are interfaces to reusable software libraries (Robillard et al. 2012). A major motivation behind LCSD is to build applications, get reviews from the users, and incorporate those changes quickly (Waszkowski 2019). Some software development approaches are quite popular and supported by different LCSD platforms, such as Iterative software development (Basil and Turner 1975) which is based on the iterative development of the application. In this way, every step is cyclically repeated one after another. In practice, this is very helpful because it allows developing and improving the application gradually. Another approach can be Rapid application development (RAD) (Beynon-Davies et al. 1999) is a software development methodology that promotes the rapid release of a software prototype. It is an agile approach and aims to utilize user feedback from the prototype to deliver a better product. Another popular methodology is the agile development methodology (Beck et al. 2001) which is a

collection of approaches and practices that promote the evolution of software development through collaboration among cross-functional teams.

Different LCSD teams may adopt different SDLC approaches. However, we focus mostly on Agile methodology for this study because Agile and LCSD can go hand in hand because the fundamental principle and objective are customer satisfaction and continuous incremental delivery. Traditional software development teams widely use agile, which also provides the generalizability for other methodologies. So, in this study, we map agile software development life cycle phases with LCSD methodologies. The inner circle of Fig. 1 shows the important development phases of an LCSD application, as outlined in (Sahay et al. 2020). The outer circle of Fig. 1 shows the phases in a traditional agile software development environment. As LCSD platforms take care of many application development challenges, some of the agile application development phases have shorter time/execution spans in LCSD than traditional software development.

3 Study Data Collection and Topic Modeling

In this Section, we discuss our data collection process to find LCSD related posts (Section 3.1). We then discuss the details about our pre-processing and topic modeling steps on the selected posts (Section 3.2).

3.1 Data Collection

We collect LCSD related SO posts in three steps: (1) Download SO data dump, (2) Identify LCSD related tag list, and (3) Extract LCSD related posts from the data dump based on our selected tag list. We describe the steps below.

Step 1: Download SO data dump. For this study, we used the most popular Q&A site, Stack Overflow (SO), where developers from diverse background discuss about various

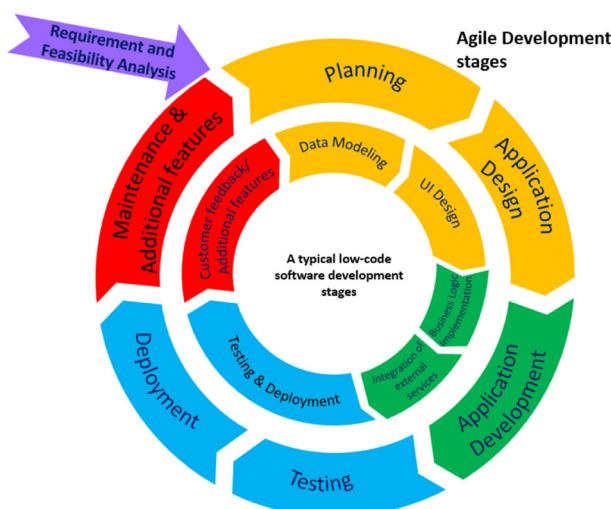


Fig. 1 Agile methodologies in traditional vs LCSD development

software and hardware related issues (2020). For this study, We downloaded SO data dump (Exchange 2020) of May 2021 which was the latest dataset available during the starting of this study. We used the contents of “Post.xml” file, which contained information about each post like the post’s unique ID, type (Question or Answer), title, body, associated tags, creation date, view-count, etc. Our data dump included discussion of 12 years from July 2008 to July 2021 and contained around 53,086,327 posts. Out of them, 21,286,478 (i.e., 40.1%) are questions, 31,799,849 (i.e., 59.9%) are answers, and 51.5% questions had accepted answers. Around 12 million users from all over the world participated in the discussions.

Each SO post contains 19 attributes, and some of the relevant are: (1) Post’s body with code snippets, (2) Post’s Id, creation and modification time, (3) Post’s view count, favorite count, score, (4) User Id of the creator, (5) Accepted answer Id and a list of 0 to 5 tags.

Step 2: Identify low-code tags. We need to identify the tags that are related to LCSD in order to extract low-code related posts from SO discussions. To find relevant tags, we followed a similar procedure used in prior work (Alamin et al. 2021; Uddin et al. 2021b; Abdellatif et al. 2020; Ahmed and Bagherzadeh 2018; Wan et al. 2019; Linares-Vásquez et al. 2013). At Step 1, we identify the initial low-code related tags and call them T_{init} . At Step 2, we finalize our low-code tag list following related work (Bagherzadeh and Khatchadourian 2019; Yang et al. 2016b). Our final tag list T_{final} contains 64 tags from the top 38 LCSD platforms. We discuss each step in details below.

- (1) Identifying Initial low-code tags. The SO posts do not have tags like “low-code” or “lowcode”. Instead, we find that low-code developers use an LCSD platform name as a tag, e.g., “appmaker” for Google Appmaker (2020). Hence, to find relevant tags, first, we compile a list of top LCSD platforms by analyzing a list of platforms that are considered as the market leaders in Gartner (Vincent et al. 2019), Forrester (Rymer et al. 2019), related research work (Sahay et al. 2020), and other online resources like PC magazine (The Best Low-Code Development Platforms 2022). Our compiled list contained 121 LCSD platforms, including all of our previous nine platforms from previous study (Alamin et al. 2021). Then for each of the LCSD platforms, we manually searched for the SO tags in SO. For example, we search for Oracle Apex via SO search engine and find a list of SO posts. We build a potential list of tags related with this platform based on manual inspection, such as “oracle” and “oracle-apex”. Then, manually examine the metadata associated with each of these tags.¹ For example, “oracle-apex” tag’s metadata says “Oracle Application Express (Oracle APEX) is a rapid Web application development tool that lets you share data and create applications. Using only a Web browser and limited programming experience, you can develop and deploy applications that are fast and secure.” and “oracle” tag’s metadata says “Oracle Database is a Multi-Model Database Management System created by Oracle Corporation. Do NOT use this tag for other products owned by Oracle, such as Java and MySQL.”. Therefore, we select the “oracle-apex” tag for Oracle Apex platform. Not all LCSD platforms have associated SO tags; thus, they were excluded. For example, OneBlink (2022) low-code platform there is no associated SO tags and thus we exclude this from our list. In order to better understand the evolution of this domain, we excluded

¹<https://meta.stackexchange.com/tags>

discontinued LCSD platforms. For example, In Jan 2020, Google announced that they would no longer release new features for Google App Maker and discontinue it by 2021 (2021) and so we excluded this platform from our list. Finally, we found 38 relevant SO tags from 38 platforms. The fifth and the first author participated in this step, and the complete list of the platforms and tags are available in our replication package.

So, our initial list contains 38 LCSD platforms such as: Zoho Creator (2022), Salesforce (2022), Quickbase (2022), Outsystems (2022), Mendix (2022), Vinyl (2022), Appian (2022), and Microsoft Powerapps (2022). We thus focus on the discussions of the above 38 LCSD platforms in SO. We find one tag per LCSD platform as the name of the platform (e.g., “powerapps” for Microsoft Powerapps platform). Thus, We refer to these 38 tags as T_{init} .

- (2) Finalizing low-code related tags. Intuitively, there might be more variations to tags of 38 LCSD platforms other than those in T_{init} . We use heuristics from previous related works (Alamin et al. 2021; Bagherzadeh and Khatchadourian 2019; Yang et al. 2016b) to find other relevant tags. First, we denote our entire SO dump data as Q_{all} . Second, we extract all the questions Q that contain any tag from T_{init} . Third, we create a candidate tag list $T_{candidate}$ using all the tags found in questions Q . Fourth, we select significantly relevant tags from $T_{candidate}$ for our LCSD discussions. Following related works (Uddin et al. 2021b; Bagherzadeh and Khatchadourian 2019; Yang et al. 2016b), we compute significance and relevance for each tag t in $T_{candidate}$ with respect to Q (our extracted questions that has T_{init} tag) and Q_{all} (i.e., our data dump) as follows,

$$(Significance) \quad S_{tag} = \frac{\# \text{ of ques. with the tag } t \text{ in } Q}{\# \text{ of ques. with the tag } t \text{ in } Q_{all}}$$

$$(Relevance) \quad R_{tag} = \frac{\# \text{ of questions with tag } t \text{ in } Q}{\# \text{ of questions in } Q}$$

A tag t is significantly relevant to LCSD if the S_{tag} and R_{tag} are higher than a threshold value. We experimented with a wide range of values of $S_{tag} = \{0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35\}$ and $R_{tag} = \{0.001, 0.005, 0.010, 0.015, 0.020, 0.025, 0.03\}$. Figure 2 shows the total number of recommended vs relevant tags from our 49 experiments. It shows that as we increase S_{tag} and R_{tag} the total number of recommend tags decreases. For example, we find that for $S_{tag} = .05$ and $R_{tag} = 0.001$ the total number of recommended tags is 61 which is highest. However, not all of the suggested tags are LCSD-related. For instance, according to our significance and relevance analysis, tags such as “oracle-
xe”, “ems”, “aura-framework”, “power-automate” etc are frequently correlated with other LCSD platform tags, although they do not contain low-code-related discussions. After manually analysing these 61 tags we find that 26 tags are relevant to LCSD-related discussions. So, for the lowest $S_{tag} = 0.3$ and $R_{tag} = 0.001$ we find 26 additional LCSD-related tags. These values are consistent with related work (Uddin et al. 2021b; Alamin et al. 2021; Bagherzadeh and Khatchadourian 2019; Ahmed and Bagherzadeh 2018). The final tag list T_{final} contains 64 significantly relevant tags. So, after combining with out initial taglist, i.e., T_{init} , our final tag list T_{final} contains 64 significantly relevant LCSD-related tags which are:

{ apex-code, lotus-notes, domino-designer-eclipse, visualforce, salesforce-chatter, apex, salesforce-service-cloud, simple-salesforce, salesforce-ios-sdk,

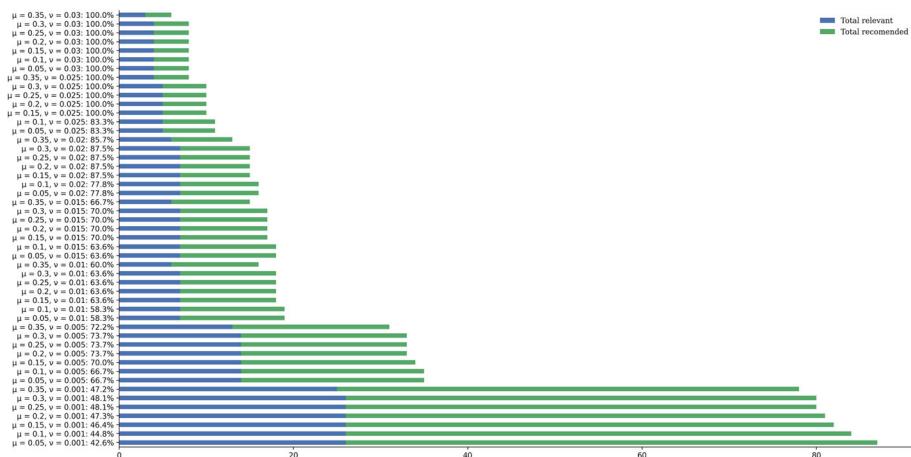


Fig. 2 Distribution of LCSD related recommended vs relevant tags based on different u and V values from our initial taglist

apex-trigger, oracle-apex-5, salesforce-lightning, salesforce-communities, oracle-apex-5.1, servicenow-rest-api, powerapps-formula, salesforce-marketing-cloud, powerapps-selected-items, powerapps-modeldriven, powerapps-collection, powerapps-canvas, oracle-apex-18.2, lwc, salesforce-development, oracle-apex-19.1, oracle-apex-19.2, outsystems, appian, quickbase, powerapps, oracle-apex, salesforce, zoho, mendix, servicenow, pega, retool, vinyl, kissflow, bizagi, neutrinos-platform, rad, joget, filemaker, boomi, opentext, tibco, webmethods, conductor, temenos-quantum, shoutem, oracle-cloud-infrastructure, amazon-honeycode, convertigo, lotus-domino, genero, genesis, gramex, processmaker, orocrm, slingr, unqork, uniface, structr }

Step 3: Extracting low-code related posts. An SO question can have at most five tags, and we consider a question as low-code related question if at least one of its tag is in our chosen tag list T_{final} . Based on our T_{final} tag set, we found a total of 27,880 questions from our data dump. SO has a score-based system (up-vote and down-vote) to ensure the questions are in proper language with necessary information (code samples and error messages), not repeated, off-topic or incorrectly tagged. Here is an example for a question with score “-4” where a practitioner is making an API related query in Powerapps ([Q61147923](#))² platform. However, it is not clear what the practitioner is asking as the question is poorly written and without any clear example. So, in order to ensure good quality discussions, we excluded questions that had a negative score. Following previous research (Uddin et al. 2021b; Bagherzadeh and Khatchadourian 2019; Rosen and Shihab 2016; Barua et al. 2014), we also excluded unanswered answers and only considered accepted answers for our dataset. Hence, our final dataset B contained 37,766 posts containing 67.4% Questions (i.e., 26,763) and 32.6% Accepted Answers (i.e., 11,010).

To ensure that our final taglist T_{final} comprises discussions relating to low-code software development, we randomly select 96 questions from our dataset that are statistically

² Q_i and A_i denote a question Q or answer A in SO with an ID i

significant with a 95% confidence level and 10 confidence interval. First and third authors contributed to this manual analysis, and after manual analysis, we found that 98% of questions from our selected taglist contain low-code platform-related discussion, with only two questions containing discussion that is not particularly related to low-code platforms. For instance, question [Q59402662](#) includes the tag “appian”, yet the question body describes only about a MySQL database performance-related issue on the Azure platform. Similarly, the question [Q19289762](#) contains the “apex-code” tag, but exclusively discusses AWS cloud authentication signature-related issues in its problem description.

3.2 Topic Modeling

We produce LCSD topics from our extracted posts in three steps: (1) Preprocess the posts, (2) Find optimal number of topics, and (3) Generate topics. We discuss the steps below.

Step 1. Preprocess the posts. For each post text, we remove noise following related works (Abdellatif et al. 2020; Bagherzadeh and Khatchadourian 2019; Barua et al. 2014). First, we remove the code snippets from the body, which is inside `<code></code>` tag, HTML tags such as (`<p></p>`, `<a>`, `` etc), and URLs. Then we remove the stop words such as “the”, “is”, “are”, punctuation marks, numbers, non-alphabetical characters using the stop word list from MALLET (McCallum 2002), NLTK (Loper and Bird 2002), and our custom low-code specific (i.e., LCSD platform names) stop word list. We remove the platform’s name from the dataset since, based on our experiments with LDA topic modeling for this study and our past work (Alamin et al. 2021), the resultant topics sometimes tend to cluster around LCSD platforms rather than the technical challenges discussed. Thus, we remove the LCSD platform names from our dataset. After this, we use porter stemmer (Ramasubramanian and Ramya 2013) to get the stemmed representations of the words e.g., “wait”, “waits”, “waiting”, and “waited”—all of which are stemmed to base form “wait”.

Step 2. Finding the optimal number of topics. After the prepossessing, we use Latent Dirichlet Allocation (Blei et al. 2003) and the MALLET tool (McCallum 2002) to find out the LCSD-related topics in SO discussions. We follow similar studies in Software engineering research using topic modeling (Arun et al. 2010; Asuncion et al. 2010; Yang et al. 2016b; Bagherzadeh and Khatchadourian 2019; Abdellatif et al. 2020). Our goal is to find the optimal number of topics K for our dataset B so that the *coherence* score, i.e., encapsulation of underlying topics, is high. We use Gensim package (Rehurek and Sojka 2010) to determine the coherence score following previous works (Uddin and Khomh 2017a; Röder et al. 2015). We experiment with different values of K that range from {5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70} and for each value, we run MALLET LDA on our dataset for 1000 iterations (Bagherzadeh and Khatchadourian 2019). Then we observe how the coherence score is changing with respect to K . We pick the topic model with the highest coherence score. Choosing the right value of K is important because, for smaller values of K , multiple real-world concepts merge, and for a large value of K , a topic breaks down. For example, in our result, the highest coherence score 0.50 for $K = 45$ and $K = 40$. The first, third, fourth, and fifth authors participate in the analysis and we choose $K = 45$ as it captures our underlying topics better. MALLET also uses two hyper-parameters, α and β , to distribute words and posts across the topics. Following the previous works (Alamin et al. 2021; Uddin et al. 2021b; Bagherzadeh and Khatchadourian 2019; Ahmed and Bagherzadeh 2018; Bajaj et al. 2014; Rosen and

Shihab 2016), we use the standard values $50/K$ and 0.01 for hyper-parameters α and β in our experiment.

Step 3. Generating topics. Topic modeling is a method of extracting a set of topics by analysing a collection of documents without any predefined taxonomy. Each document has a probability distribution of topics, and every topic has a probability distribution of a set of related words (Barua et al. 2014). We produced 45 topics using the above LDA configuration on our dataset B . Each topic model offers a list of top N words and a list of M posts associated with the topic. In our settings, a topic consists of 30 most frequently co-related words, which represent a concept. Each post had a correlation score between 0 to 1, and following the previous work (Alamin et al. 2021; Uddin et al. 2021b; Wan et al. 2019), we assign a document with a topic that it correlates most.

4 Empirical Study

We report the results of an empirical study by answering the following five research questions (RQ) based on our analysis of LCSD topics in our dataset.

- RQ1.** What topics do LCSD practitioners discuss? (Section 4.1)
- RQ2.** How do the LCSD topics evolve over time in SO? (Section 4.2)
- RQ3.** What types of questions are asked across the observed topic categories? (Section 4.3)
- RQ4.** How are the observed topic categories discussed across SDLC phases? (Section 4.4)
- RQ5.** What LCSD topics are the most difficult to answer? (Section 4.5)

The first two research questions (RQ1, RQ2) provide insights about what topics practitioners discuss in SO and how these topics have evolved over time. The third and fourth research questions (RQ3, RQ4) explore the types of questions in these topics and they affected different SDLC phases. At the end, we discuss the popularity and difficulty of the LCSD topics in the last research question (RQ5).

4.1 What Topics are Discussed About LCSD in Stack Overflow? (RQ1)

4.1.1 Motivation

The increased popularity of LCSD as a flexible and straightforward approach helps develop practical business applications. The challenges of LCSD are yet to be studied as this is a new approach to software development. SO is an established source of knowledge repository to systematically study the real-world challenges that the practitioners face. An understanding of the LCSD topics in SO developer discussions will help LCSD platform providers and researchers to have a better understanding of the underlying prevalent issues, which can then help guide efforts to improve the quality of LCSD platforms.

4.1.2 Approach

We applied LDA topic modeling to our LCSD-related discussion in SO. We get 45 low-code related topics from our LDA topic modeling, as discussed in Section 3. We use card sorting (Fincher and Tenenberg 2005) to label these topics following previous works (Bagherzadeh and Khatchadourian 2019; Ahmed and Bagherzadeh 2018; Yang et al. 2016b; Rosen and Shihab 2016; Abdellatif et al. 2020). In open card sorting, there is no predefined

list of labels. Following related works (Alamin et al. 2021; Uddin et al. 2021b; Bagherzadeh and Khatchadourian 2019; Abdellatif et al. 2020), we label each topic by analyzing the top 30 words for the topic and a random sample of at least 20 questions that are assigned to the topic. Four of the authors participated in the labeling process in group sessions (first, third to fifth). Each author assigns a label to each topic and discusses it until there is an agreement. The authors reached an agreement after around 10 iterations of meetings over Skype and email and labeled the 45 topics from the LDA output.

After this initial labeling, we merged a few topics because they contained similar discussions with different vocabularies. For example, we merged topic 36 and 43 into Dynamic form controller because both topics contained discussions related to forms with a predefined list of values, dynamically changing the fields (or options) of forms values based on users' actions or previous selections. Similarly, we merged topic 2 and 19 to DB Setup & Migration. In the end, we obtained 40 distinct LCSD-related topics.

After the labeling of the topics, we revisited the labels in an attempt to find any clusters/groups among the topics. For example, Date & Time Manipulation, Formatted Data Parsing, and Pattern Matching topics are related, and thus, they are grouped under the General Programming category. We repeated this process multiple times to find increasingly higher categories. For example, we found another category called Dynamic Content which contained two topics Dynamic Data Binding and Dynamic Data Filtering. We then put these two categories under called Business Logic Implementation. This higher abstraction helped us to place other topics related to implementing business logic under this category. Following a similar strategy, we put this Business logic implementation under the Customization category, which discussed customizing applications. For example, under Customization, there were a category called UI which contained Dynamic Layout, and Script category, which contained topics such as Dynamic Page Layout, Dialog Box Manipulation, Window Style Manipulation, and Dynamic Form Controller. The entire process of creating this hierarchy of topic categories took multiple iterations and revisions. We created a coding guideline for creating the taxonomy of topics to ensure consistency and reproducibility. We share the coding guide with our replication package.

4.1.3 Results

After labeling and merging, we find 40 LCSD-related topics. Then after grouping these topics into higher categories, we find five high-level categories: (1) Customization, (2) Data Storage, (3) Platform Adoption, (4) Platform Maintenance, and (5) Third-Party Integration. Figure 3 shows the distribution of topics and questions into these five categories. Among these categories, Customization has the highest coverage of questions and topics (30% Questions in 11 Topics), followed by Data Storage (25% Questions in 9 Topics), Platform

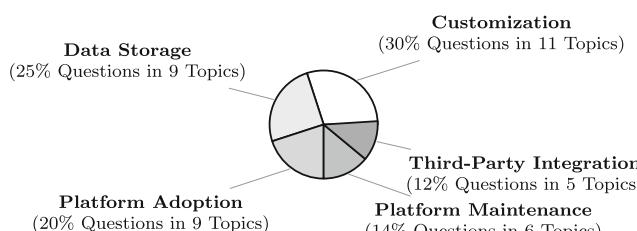


Fig. 3 Distribution of questions and topics per topic category

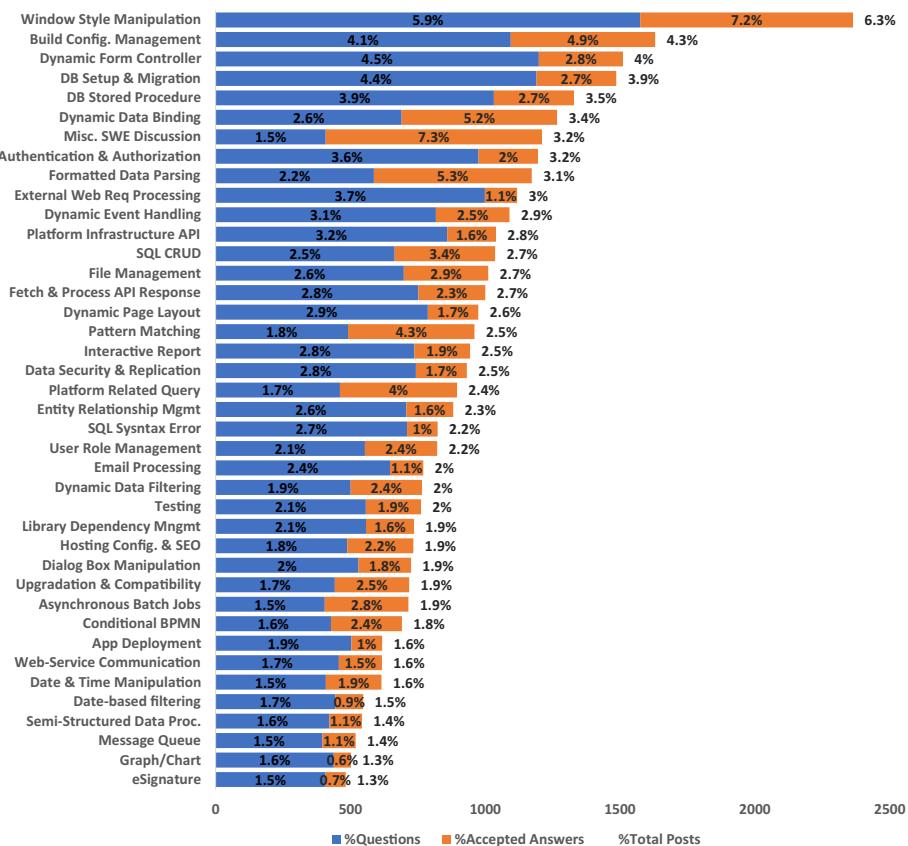


Fig. 4 Distribution of questions by low-code topics by total number of posts (Number of Questions + Number of Accepted Answers)

Adoption (20% Questions in 9 Topics), Platform Maintenance (14% Questions in 6 Topics), Third-Party Integration (12% Questions in 5 Topics).

Figure 4 shows the 40 LCSD topics sorted by numbers of posts. A post means an LCSD-related question or an accepted answer in our case. As discussed in Section 3.1, our dataset has total 37,773 posts containing 26,763 questions and 11,010 accepted answers. The topic with the highest number of posts is placed on top of the list. On average, each topic has 944 posts (question + accepted answer). The topic Window Style Manipulation has the highest number of posts (6.3%) with Questions 5.9% of total questions and 7.2% total accepted answers. On average, each topic has around 669 questions.

Figure 5 provides a taxonomy of 40 LCSD related topics into five categories. The topics are organized in descending order of the number of questions. For example, the Customization category has the highest number of questions, followed by Data Storage. Each category may have some sub-categories of topics. For example, the Customization category has two sub-categories: UI and Business Logic. The topics under each sub-category can further be grouped into multiple sub-sub-categories. For example, the UI sub-category has 4 topics grouped into Script and Dynamic Layout sub-sub-categories. Each sub-category,

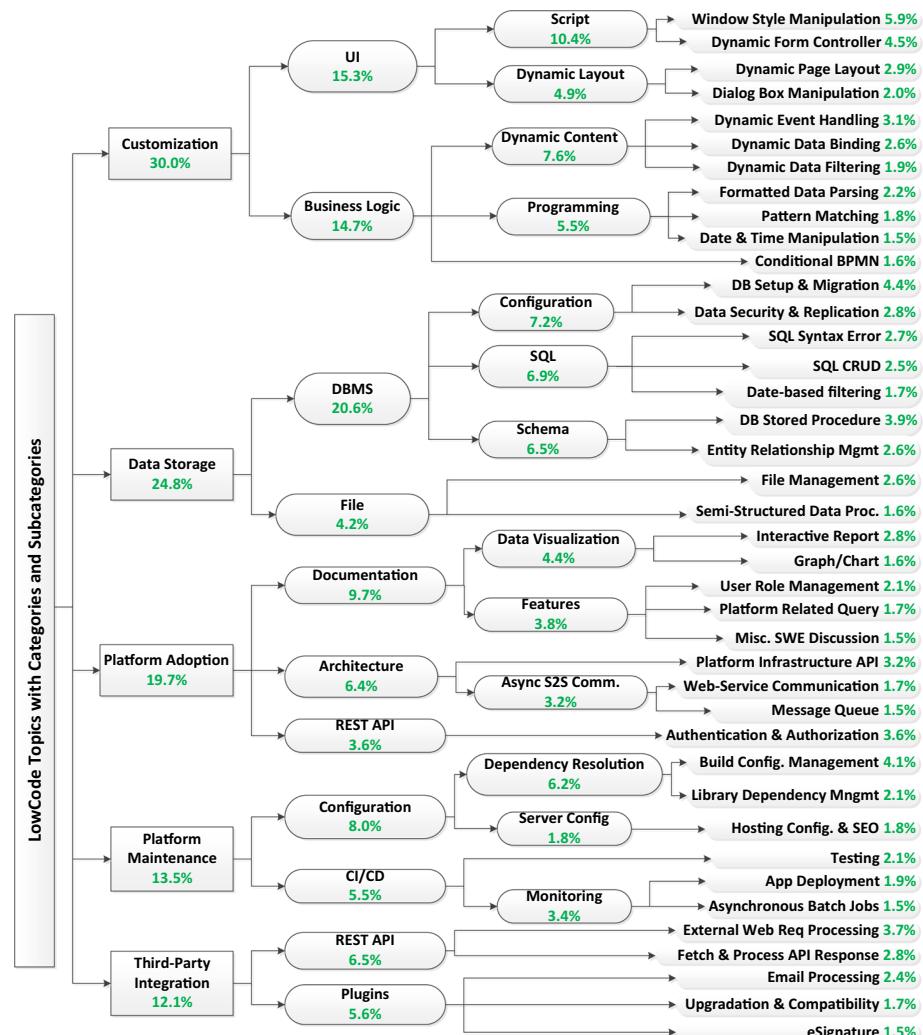


Fig. 5 A taxonomy of LCSD topics with sub-categories

sub-sub-categories, and topics are also presented in descending order of the number of questions.

We discuss the five categories and the 40 topics in detail below.

Customization Topic Category. Customization is the largest topic category in terms of the number of topics and percentage of questions. Out of the 40 topics, 11 topics belong to the Customization category, with around 30% of questions in our dataset. These topics contain discussions about implementing business logic, customizing UI, input and form data validation, general programming-related query to implement some features, etc. This category has two sub-categories: (1) UI contains discussion about customizing the UI, dynamically changing window components, interactive dialog boxes, and (2)

Business Logic contains discussion about different programming customization-related queries, dynamically binding UI elements to backend data.

• **UI Sub-Category** contains 15% questions and four topics divided in two sub-sub-categories: (1) Script contains discussion about manipulation of text widgets, formatting components, and (2) Dynamic Layout is about hiding and moving components, showing popups.

The **Script** sub-sub-category contains 10.4% questions and has two topics: (1) Topic *Window Style Manipulation* (5.9%) concerns about manipulating the style of the HTML documents such as adding/removing margins/padding (e.g., [Q36503030](#)), adding links, manipulating navigation bar and embedded views (e.g., [Q30453620](#)). (2) Topic *Dynamic Form Controller* (4.5%) are about dynamic form, i.e., forms with predefined list of values (e.g., [Q64373454](#)), multi select content (e.g., [Q39318510](#)), changing forms option based on previous selection (e.g., [Q43725028](#)). The **Dynamic Layout** sub-sub-category covers 4.9% questions and has two topics: (1) Topic *Dynamic Page Layout* (2.9%) contains discussion about UI (i.e. page) customization (e.g., [Q65964413](#)), pagination in [Q4536018](#), hiding or moving element based on some user action or an event (e.g., [Q13231072](#)), (2) Topic *Dialog Box Manipulation* (2.0%) is about manipulating dialog box (e.g., pop up/modals) such as hiding them in [Q49804455](#), close them in [Q55513527](#), show popup, refresh web-page (e.g., [Q60606986](#), [Q21701437](#)).

• **Business Logic Sub-Category** contains 14.7% questions and 7 topics in two sub-sub-categories: (1) Programming is about discussion related to different programming-related questions and data access, and (2) Dynamic Content is about discussions related to dynamically querying data from different data sources, dynamically changing the web-page content. The Business Logic sub-category contains one topic *Conditional BPMN* that does not belong to any sub-sub-category. Topic *Conditional BPMN* (1.6%) contains LCSD platform's application customization related discussions on Business Process Model and Notation (BPMN) (e.g. [Q38265464](#)) and conditional logic features (e.g., [Q66335289](#), [Q65838553](#)).

The **Dynamic Content** sub-sub-category contains 7.6% of questions and has three topics: (1) Topic *Dynamic Event Handling* (3.1%) discusses about different JavaScript related issues such as JavaScript feature not working (e.g., “JavaScript promise is not working” in [Q65550370](#)), browser compatibility, JS event initialization issue (e.g., [Q64507615](#)) etc. (2) Topic *Dynamic Data Binding* (2.6%) is about discussions related to the design of forms with predefined values (e.g., [Q45051098](#)), the implementation of multi-select, customized drop-down values, form validation (e.g., [Q51115573](#)), changing content of one field based on the value of other field in [Q47652192](#). (3) Topic *Dynamic Data Filtering* (1.9%) contains business logic customization related discussion based on advanced filtering criteria and querying multiple tables. (e.g., “Find Records Based on the Contents of Records in a Related Table?” in [Q20665253](#) and “find the latest record grouped by name in layout” in [Q38128584](#)).

The **Programming** sub-sub-category contains 5.5% of questions and has three topics: (1) Topic *Formatted Data Parsing* (2.2%) is about programming related discussion on parsing formatted data, i.e., JSON (e.g., [Q50184058](#), [Q44803257](#)), XML (e.g., [Q13785513](#)), array of objects (e.g., [Q66744874](#)) etc. (2) Topic *Pattern Matching* (1.8%) topic concerns programming related discussions about searching and modifying strings by pattern matching using regular expression (e.g., “How do I search for an Exact phrase” in [Q51258323](#), “Regex pattern to replace html in a given text string” in [Q48251198](#)). (3) Topic *Date & Time Manipulation* (1.5%) contains programming discussions about date-time

manipulation like conversion of formatted string from data-time in [Q51714301](#), calculation of difference between date-time (e.g., [Q59230493](#)) and timezone, time conversion (e.g., “How to convert a Date value to unix timestamp?” in [Q60601201](#)).

Data Storage Topic Category. Data Storage Category is the second largest topic category with a total of 9 topics and around 25% of the questions of our dataset. This topic category contains discussions on database management and file storage. It contains two sub-categories: (1) DBMS is about discussion related to database setup, migration, DB query, (2) File concerns storing and retrieving files (i.e., images, CSV files, etc.).

- **DBMS Sub-Category** contains around 20.6% questions with seven topics under three sub-sub-categories: (1) Configuration contains discussions about database setup, database connection, DB data security, (2) SQL contains discussion about SQL query, (3) Schema is about database schema design (i.e., Primary key, foreign key design), different stored procedure.

Configuration sub-sub-category contains 7.2% questions and two topics: (1) Topic *DB Setup & Migration* (4.4%) topic is about connecting applications to different vendor databases (i.e., MySQL, Postgres, Oracle etc.) (e.g., “is ODBC Firebird connection possible?” in [Q28251836](#)), DB users (e.g., [Q58815776](#)), issues about different database versions, data migration to LCSD platform (e.g., “How to add External data source into MySQL?” in [Q28251836](#) or [Q22626970](#)). (2) Topic *Data Security & Replication* (2.8%) topic concerns about discussion related to data security (e.g., encryption and decryption in [Q1567252](#)), accessing stored of database file (e.g., [Q5730482](#)), data backup or replication (e.g., [Q10997987](#)) etc.

SQL sub-sub-category contains 6.9% questions and three topics: (1) Topic *SQL Syntax Error* (2.7%) discusses about errors in syntax in different SQL query and stored procedure. For example, there are questions such as “I’m getting error while creating a procedure in pl/sql” in [Q63990287](#), “SQL parsing fails, not sure what the issue is?” in [Q8298486](#). (2) Topic *SQL CRUD* (2.5%) is about database Create, read, update and delete (i.e., CRUD) related queries (e.g., [Q22712624](#)), and advanced queries too, such as inner join, nested join, aggregate (e.g., “SQL Query: JOIN Three tables then Not showing the results after joining the 3rd table” in [Q22712624](#)). This topic also contains discussion about Object query language, which is a high-level wrapper over SQL in [Q64812548](#). (3) Topic *Date-based filtering* (1.7%) contains database query related discussion specially for date-time based filtering (e.g., [Q52389335](#)), i.e., monthly/quarterly query, time-based data grouping etc. For example, there are questions such as How to count total amount of value by day (e.g., [Q65142062](#)).

Schema sub-sub-category contains 6.5% questions and two topics: (1) Topic *DB Stored Procedure* (3.9%) is about database schema and advanced database related discussion on stored procedure (e.g., support of triggers in LCSD platform, [Q11799577](#), [Q37810803](#)). (2) Topic *Entity Relationship Mgmt* (2.6%) concerns about discussion on advanced database schema design (e.g., “How to automatically insert foreign key into table after submit in [LCSD platform]”, [Q66968187](#)) and database discussion to automatically update database (e.g., “Auto increment item in Oracle APEX” in [Q61961348](#)).

- **File Sub-Category** contains 4.2% questions with two topics: (1) Topic *File Management* (2.6%) contains discussion of file management, i.e., storing and processing files, renaming it in [Q56414466](#), converting files from one format to another in [Q45796962](#), handling image files (e.g., [Q34203211](#)). (2) Topic *Semi-Structured Data*

Proc. (1.6%) is about different programming related discussion on processing, modifying and storing semi-structured data files, i.e., XML, CSV files. For example, there are questions such as Fetch CSV file columns dynamically Using [platform] package in [Q66310875](#).

Platform Adoption Topic Category. A total of nine topics belong to the Platform Adoption category with around 20% questions. The nine topics belong to three sub-categories: (1) Documentation contains LCSD platform's feature-related discussions and how to use those features, (2) Architecture concerns about what type of software development architecture (e.g., client-server communication) is supported by the LCSD platforms, (3) REST API contains LCSD platform's RESTful APIs.

- **Documentation Sub-Category** contains 9.7% questions and five topics. Four of the topics fall under two sub-sub categories: (1) Data Visualization contains discussion related to interactive reports and graphs, (2) Features is about LCSD platform provides features such as user's role management, support on SDLC management. Topic *Misc. SWE Discussion* (1.5%) concerns about discussions related to general software engineering such as Unix Threading (e.g., [Q30530873](#)), Object-oriented programming (e.g., [Q314241](#)), auto scaling, ambiguous documentation in [Q10348746](#).

Data Visualization sub-sub-category contains 4.4% questions and has two topics: (1) Topic *Interactive Report* (2.8%) is about data visualization and interactive data reports. It contains developers' discussions about how they can use different platform features for customized reports. For example, "using jquery hide column heading when no data in column in interactive report in [platform]" in [Q53294659](#). (2) Topic *Graph/Chart* (1.6%) discusses about platform's support and documentation request to draw different graphs (e.g., [Q41257691](#)) and charts using stored data. For example, "How to overlay a line plot over a bar graph in [platform]?" in [Q28727869](#)

Features sub-sub-category contains 3.8% questions and has two topics: (1) Topic *User Role Management* (2.1%) contains discussion about different user role management features (i.e., administrators and regulators) provided by the LCSD platforms. It discusses about user's profile management (e.g., [Q66853056](#)), user group and access-level (e.g., [Q35457840](#)). (2) Topic *Platform Related Query* (1.7%) contains general discussion about LCSD platforms such as comparison of features between different platforms (e.g., "How is [platform A] better than [platform B] in BPM?" in [Q39127918](#)), Agile and RAD development support (e.g., [Q2512396](#)), performance of a specific feature of a platform in [Q6068882](#).

- **Architecture Sub-Category** contains 6.4% questions and two topics and one sub-sub category: (1) Async S2S Comm contains discussion related to distributed applications with service to service communication. Topic *Platform Infrastructure API* (3.2%) contains cloud-based REST API from the LCSD platforms to configure and utilize different platform features, connect to other services or data sources via connectors or cloud REST APIs. For example, the questions are about how [platform] apps portal integration with [DB] On-premise in [Q63688934](#), "Change Shape OCI instance with Ansible" in [Q60511836](#).

Async S2S Comm sub-sub-category contains 3.2% questions and has two topics: (1) Topic *Web-Service Communication* (1.7%) contains discussions about micro-service architecture, service to service communication via web service description language (e.g., [Q16278661](#), [Q2567466](#)), HTTP REST message in [Q58689313](#), Windows Communication Foundation (e.g., [Q36849686](#)). (2) Topic *Message Queue* (1.5%) is about discussion about different asynchronous service-to-service data exchange

mechanisms such as using a message queue. It generally contains discussions about micro-service design patterns and producer and consumer mechanisms (e.g., [Q41640881](#)) for data exchange. For example, “How to know who is connected to a [Platform] EMS Queue” in [Q66999418](#), [Q56334001](#).

REST API sub-category contains 3.6% questions and has one topics: (1) Topic *Authentication & Authorization* (3.6%) contains discussion about LCSD platforms support on different standard authentication and authorization protocol such as OAuth2 (e.g., [Q30475542](#)), SAML (e.g., [Q23624206](#)), access token (e.g., “access token in android [Platform] sdk” in [Q32943204](#)).

Platform Maintenance Topic Category. We find 6 topics and 13.5% questions in Platform Maintenance category. It has two sub-categories: (1) Configuration contains discussion on LCSD platforms library and build configuration, (2) CI/CD. is about discussion related to DevOps tasks such as continuous integration and continuous delivery, testing etc.

- **Configuration Sub-Category** contains 8.0% questions and three topics under two sub-sub categories: (1) Dependency Resolution is about LCSD platforms server’s library dependency management, (2) Server Configuration is about LCSD platform’s servers configuration and hosting settings such as SSL configuration.

Dependency Resolution sub-sub-category contains 6.2% questions and has two topics: (1) Topic *Build Config. Management* (4.1%) contains discussion about system build configuration and external library management-related issues in [Q48727452](#). This topic also contains discussion about compilation failure (e.g., [Q29243987](#)), library dependency, build path not configured properly (e.g., [Q57015131](#)) etc. (2) Topic *Library Dependency Mngmt* (2.1%) is about the library and dependencies of the system (e.g. [Q23471590](#), [Q62872836](#)), server configuration, different library version compatibility issues in [Q60050869](#). **Server Config.** sub-sub-category contains 1.8% questions and has one topic: (1) Topic *Hosting Config. & SEO* (1.8%) is about discussions about LCSD platforms support on server configuration, i.e., configuring SSL certificate (e.g., “[platform] client ignoring expired certificate” in [Q55044903](#)), LCSD platform’s support on making the application accessible and index-able (e.g., [Q34860991](#)).

- **CI/CD Sub-Category** contains 5.5% questions and three topics. Two of the topics fall under one sub-sub category: (1) Monitoring is about the discussion on monitoring the deployed applications and scheduled job status. Topic *Testing* (2.1%) contains discussions about LCSD platforms support on testing and test coverage. For example, “How to know overall code coverage of multiple test classes?” in [Q67724447](#), How to write a test class as in [Q50586482](#).

Monitoring sub-sub-category contains 3.4% questions and has two topics: (1) Topic *App Deployment* (1.9%) discusses about the LCSD platform’s CI/CD features such as incrementally updating the application code in [Q39045129](#), deployment packages as in [Q4813597](#), monitoring the changes in the application code (e.g., [Q61938011](#)). (2) Topic *Asynchronous Batch Jobs* (1.5%) contains discussions about LCSD platforms’ support for monitoring and scheduling asynchronous batch jobs and scheduled tasks. For example, “How to get your failing batch records?” in [Q11068830](#), “How can I schedule apex to run every 30 seconds?” in [Q17143633](#).

Third-Party Integration Topic Category is smallest topic category based on number of questions (12.1% questions). It has five Topics. Four of its topics fall under two sub-categories: (1) REST API contains discussion related to RestFul API communication with third-party services, (2) Plugins is about discussion about external plugins and APIs that are supported by the LCSD platforms.

- **REST API Sub-Category** contains 6.5% questions and two topics: (1) Topic *External Web Req Processing* (3.7%) contains discussion about integrating 3rd party REST APIs, processing and parsing external requests such as “Connect to [Platform] REST API with [Service] data integration” in [Q51865601](#), [Q46033973](#). (2) Topic *Fetch & Process API Response* (2.8%) contains discussions about making HTTP request to remote servers (e.g., “REST http post method—what does -d mean in a curl?” in [Q65877037](#)), analyzing and processing the response, handling web security issues (e.g., CORS policies in [Q60270574](#)).
- **Plugins Sub-Category** contains 5.6% questions and three topics: (1) Topic *Email Processing* (2.4%) discusses about processing automating emailing in [Q65626477](#), sending formatted HTML email (e.g., “Send HTML email using [platform]” in [Q41546887](#)), forwarding emails in [Q18234790](#) etc. (2) Topic *Upgradation & Compatibility* (1.7%) contains discussion about application version migration as in [Q16894245](#), upgradation and compatibility issues of different plugins used in low-code applications (e.g., [Q18231293](#), [Q49017642](#)). (3) Topic *eSignature* (1.5%) contains discussion about different issues and customization for electronic signature of documents, i.e., docusign about collecting user’s agreement/permission for sales or account opening. For example, there are questions such as “Auto Add Document to DocuSign [Platform] Using Custom Button” in [Q34804072](#), [Q27512874](#).

RQ1. What topics are discussed about LCSD in SO? We found 40 Topics organized into five high-level categories. The Customization category (30%) has the highest number of questions, followed by Data Storage (25%), Platform Adoption (20%), Platform Maintenance (14%), and Third-Party Integration (12%). Window Style Manipulation from the Customization category has the highest number of questions (5.9%) followed by build Configuration (4.1%) Management from the Platform Maintenance category. Our studies reveal that low-code practitioners struggle with RESTful API Integration, configuration and maintenance of the platforms. We also observed that proper documentation could have mitigated these challenges to a great extent.

4.2 How do the LCSD Topics Evolve over Time? (RQ2)

4.2.1 Motivation

Our analysis of RQ1 finds that LCSD topics are diverse. For example, the Customization topic category contains discussions about developing and customizing the application, and Platform Adoption and Platform Maintenance topic contains discussions related to different features provided by the LCSD platform providers. The platforms for LCSD continue to evolve, as do the underlying topics and question types. We study the evolution of these topics and question types to understand better the evolution and adoption of LCSD development

and its community. This analysis will provide valuable insights into the LCSD community and help identify if any topic needs special attention.

4.2.2 Approach

Following related studies (Uddin et al. 2021b), we study the absolute and relative impacts of each of our observed five LCSD topic categories as follows.

Topic Absolute Impact. We apply LDA topic for our corpus C and get K topics (t_1, t_2, \dots, t_K). The absolute impact metric for a topic t_k in a month (m) is defined as:

$$Impact_{absolute}(t_k; m) = \sum_{p_i=1}^{D(m)} \theta(p_i; t_k) \quad (1)$$

Here the $D(m)$ is the total number of SO posts in the month m and $\theta(p_i; t_k)$ denotes the possibility for a post (p_i) belonging to a topic t_k .

From our topic modeling, we found 40 topics that were categorized into five high topic categories, i.e., *Customization, Data Storage, Platform Adoption, Platform Maintenance, Third-Party Integration*. Now, we further refine the equation for absolute impact for LCSD topics to find absolute impact metrics for a topic category (TC_j) for a month m as follows:

$$Impact_{absolute}(TC_j; m) = \sum_{t_k}^{TC_j} Impact_{absolute}(t_k; m), 0 < j < TC \quad (2)$$

Topic Relative Impact. Related impact metric signifies the proportion of posts for a particular LCSD topic t_k relative to all the posts in our corpus C for a particular month m . Following related studies (Uddin et al. 2021b), we compute the related impact metrics for LCSD topics. We use the following equation to compute the metric for a topic t_k in a month m as follows:

$$Impact_{relative}(t_k, m) = \frac{1}{|D(m)|} \sum_{p_i=1}^{\theta} (p_i; t_k), 1 \leq i \leq C \quad (3)$$

Here $D(m)$ denotes the total number of posts that belongs to a topic t_k for a particular month m . Here δ denotes the probability of a particular post p_i for our Corpus C belonging to a particular topic t_k .

Similar to the absolute impact, we refine the equation to compute the relative impact on LCSD topic categories as follows:

$$Impact_{relative}(TC; m) = \sum_{t_k}^{TC} Impact_{relative}(t_k; m) \quad (4)$$

Here TC donates one of our five topic categories and topics that belong to each topic category.

4.2.3 Result

Figure 6 depicts the progression of overall LCSD-related conversation using the absolute impact equation from our extracted dataset between 2008 to 2021. Additionally, it demonstrates that the peaks of LCSD-related talks occurred in mid-2020 (i.e., around 400 questions

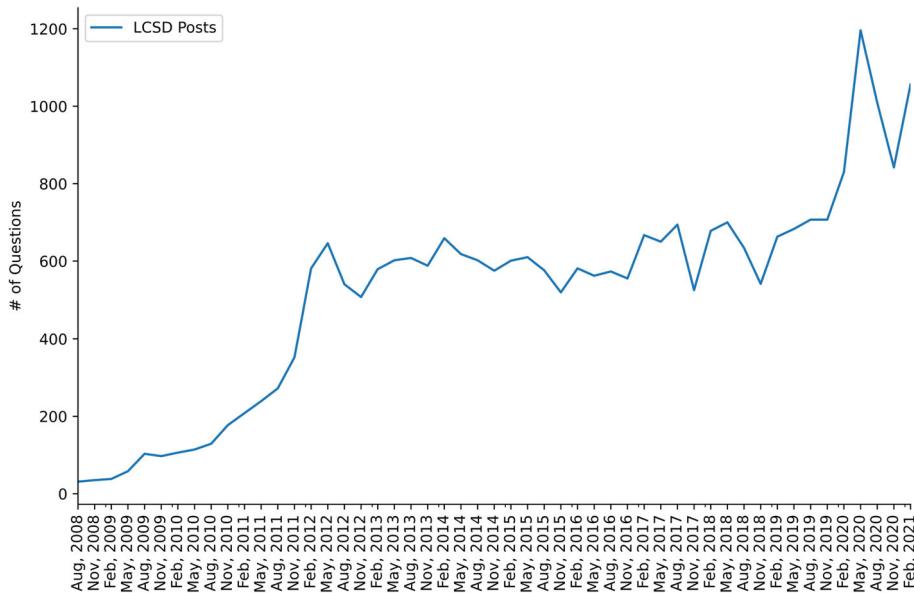


Fig. 6 The evolution of overall LCSD-related discussions over time

per month). It also reveals that LCSD-related discussion has been gaining popularity since 2012. In the section below, we provide a more extensive explanation for the minor spikes in the Fig. 6.

We observe that in the early days (i.e., 2009), Platform Adoption along with Data Storage topic category has more questions compared to Customization. Customization topic category starts to get a dominant position from mid (i.e., August) of 2011 over Platform Adoption, and it remains the dominant topic till the end of 2021. The number of questions in the Customization topic category gradually increased over time, from August 2011 (23) to March 2012 (81) to May 2020 (99). Data Storage topic category briefly exceeds Customization Category during August 2013, but it mostly holds a dominant position other times compared to Platform Adoption topic category. On the other hand, Platform Maintenance and Third-Party Integration exhibits very similar evolution over the whole time.

We further look into the Fig. 7 and see there are mainly two significant upticks in the number of posts about LCSD. The first one is between August 2011 to May 2012, when there is a sharp increase in the number of questions for almost all topic categories, especially for Customization and Data Storage Category. By this Salesforce (2022) LCSD platform-related discussion becomes quite popular in SO, and around that time, it ranks very high as a CRM platform. The second increase in posts is between February 2020 and August 2020. During this time of the Covid19 pandemic, many businesses began to operate online and there is a significant uptick in the number of questions in the Customization category, followed by Data Storage and Platform Adoption. Moreover, there is an uptick in the number of questions on building a simple app to collect and report data, especially the Salesforce platform. There is an increase in the Platform Adoption topic category between mid-2016 to mid-2017. During this time Oracle Apex platform released version 5.0, and there is an uptick of questions regarding different features such as interactive grid in [Q43316233](#), drag

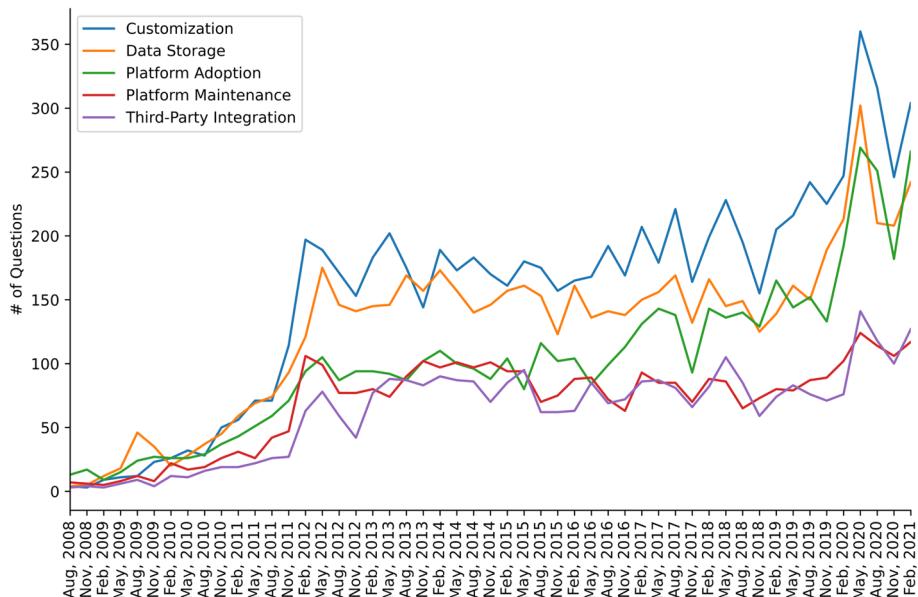


Fig. 7 The absolute impact for LCSD topic categories over time

and drop layout in [Q45818292](#). Now we provide a detailed analysis of each of the five topic categories.

Customization This is the biggest topic category with 11 topics. From 2008 to mid of 2011, all of these topics evolve homogeneously. From the mid of 2011 to the first Quartile of 2012, Dynamic Page Layout topic becomes dominant. “How to get fields in page layout” in [Q7256190](#), issues with page layout in different LCSD platforms (e.g., [Q7421985](#)). From the end of 2012 to 2017, Window Style Manipulation topic remains most dominant. “Passing values from child window to parent window which is on another domain?” in [Q16463602](#), view related issues [Q15715645](#). From the end of 2017 to the end, our dataset Dynamic Form Controller topic remains the most dominant.

Data Storage Category From mid-2015, Database Setup & Migration topic becomes the most dominant topic in this category and has some high spikes during the pandemic and mid of 2017. For instance, there are queries like “Using Jenkins for OCI database migration” in [Q62217796](#) and “Almost all the cloud service providers have 99.95% of data available on the cloud. What will happen if the whole region sinks in an earthquake?” in [Q62102679](#). Since 2017 DevOps and database “Domino Xpage database building automation or continuous integration using Jenkins with maven.” in [Q43092239](#). From mid-2011 to mid-2014, DB Stored Procedure topic remains dominant. “Oracle APEX: Call stored procedure from javascript” in [Q20501834](#).

Platform Adoption Category From 2008 to mid-2011, Platform Adoption related topics were the most dominant (e.g., “Suggested platform/tools for rapid game development and game prototyping” in [Q312357](#)). Between mid-2011 to mid-2017, Authentication & Authorization topic becomes dominant (e.g., “Can I implement my own authentication process in force.com or it is against terms of service?” [Q13059568](#), [Q13034866](#)). Since the end of 2017, Platform Infrastructure API remains the most dominant. So, practitioners

ask queries like “VirtualBox VM changes MAC address after imported to Oracle Cloud Infrastructure” in [Q61501108](#) and “How to send a classic report as mail body in oracle Apex 19.2” in [Q59693984](#), report layout ([Q59833909](#), [Q59752159](#)).

Platform Maintenance Topic Category. From 2008 to mid-2019, the Build Configuration Management topic remains the most dominant topic. It has some high spikes in the number of questions during the beginning of 2012 and the first quartile of 2014. Build error [Q21720165](#), [Q21326163](#), build projects automatically [Q21758244](#). From mid-2019, Library Dependency Management topic-related questions became popular (e.g., library-related issues (e.g., [Q62825046](#), [Q61100705](#)), library not found [Q61911916](#)).

Third-Party Integration Topic Category. The five topics from this category evolve simultaneously. From the beginning of 2015, the External Web Request Processing topic has become more dominant than other topics with a slight margin. External Web Request Processing and Fetch & Process API response, E-signature topics become dominant during the pandemic with queries such as platform support on e-signature [Q62417381](#) and etc.

In Fig. 8, we now provide more insight into the evolution of LCSD topic categories. It confirms the findings presented in Fig. 7 and adds some previously unknown insights. For instance, in the last quartile of 2009, it is apparent that Data Storage is the most popular Topic Category. According to the absolute impact metric, all five themes are increasing monotonically. The relative impact measure, on the other hand, indicates that the Customization, Platform Maintenance, and Third-Party Integration Topic group evolves in a nearly identical manner. However, this Figure demonstrates that, beginning in 2016, Platform Adoption-related conversation increased and eventually surpassed Data Storage-related discussion. This in-depth examination of evolution is significant because it demonstrates that, while Data Storage Topics are the second-largest Topic category,

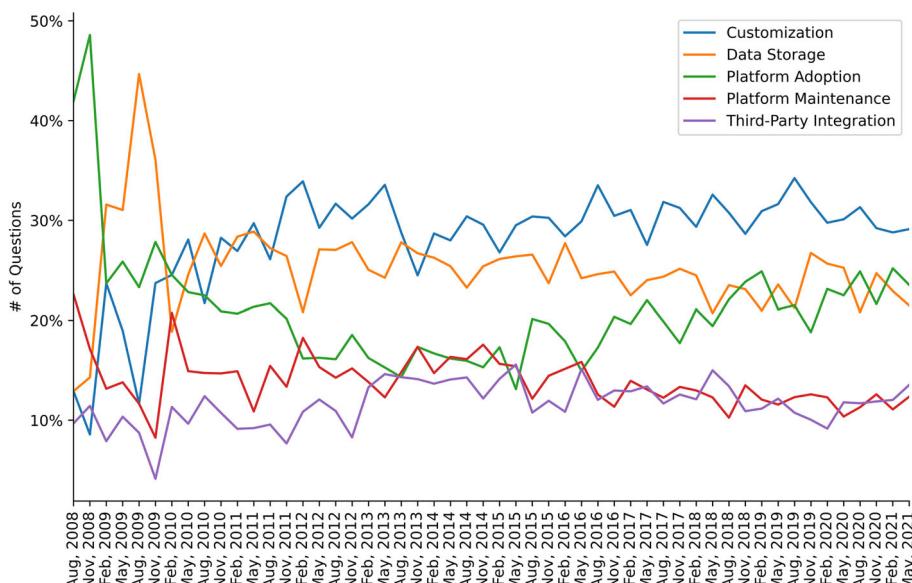


Fig. 8 The relative impact for LCSD topic categories over time

Platform Adoption-related queries are evolving rapidly and require further attention from platform vendors.

RQ2. How does the LCSD-related discussion evolve? Since 2012, LCSD-related talks in SO have grown in popularity, and this trend has accelerated since 2020. Initially, the Customization and Data Storage Topic Categories dominated, but in recent years, Platform Adoption-related inquiries have grown in popularity.

4.3 What Types of Questions are Asked Across the Observed Topic Categories? (RQ3)

4.3.1 Motivation

This research question aims to provide a deeper understanding of LCSD-related topics based on the types of questions asked about the LCSD platforms in SO. For example, “what” types of questions denote that developers are not sure about some specific characteristics of LCSD platforms, while “how” types of questions denote that they do not know how to solve a problem using an LCSD platform. Intuitively, the prevalence of “what” types of questions would denote that the LCSD platforms need to better inform the services they offer, while the prevalence of “how” type of questions would denote that the LCSD platforms need to have better documentation so that developers can learn easily on how to use those. Initially, in 2011 Treude et al. (2011) investigated different types of questions on stack overflow. Later Rosen and Shihab (2015) conducts an empirical study like ours on Mobile developers’ discussion in stack overflow with these four types of questions. Later, very similar studies on chatbot development (Abdellatif et al. 2020) and IoT developers’ discussion on Stack overflow (Uddin et al. 2021b) also explore this research question to provide more insights about specific domains and complement the findings of topic modeling.

4.3.2 Approach

In order to understand what-type of questions are discussed in SO by the LCSD practitioners, we take a statistically significant sample from our extracted dataset and then manually analyze each question and label them into one of four types: How-type, Why-type, What-type, Others-type following related studies (Uddin et al. 2021b; Abdellatif et al. 2020; Rosen and Shihab 2015). So, our approach is divided into two steps: Step 1. We generate a statistically significant sample size, Step 2. we manually analyze and label them.

Step 1. Generate Sample. As discussed in Section 3.1 our final dataset has 26,763 questions. A statistically significant sample with a 95% confidence level and five confidence intervals would be at least 379 random questions, and a 10 confidence interval would have a sample size of 96 questions. A random sample represents a representative for the entire dataset, and thus this could miss questions from the subset of questions that may belong to smaller topic categories. For example, as discussed in RQ1, we have 40 topics organized into five categories. As random sampling is not uniform across the topic categories, it might miss important questions from smaller topic categories such as Third-Party Integration. Therefore, following previous empirical studies (Uddin et al. 2021b; Abdellatif et al. 2020), we draw a statistically significant random sample from each of the five topic categories. Specifically, we draw the distribution of questions in our sample from each of the 5 topic categories with a 95% confidence level and ten confidence

intervals. The sample is drawn as follows: 95 questions from the Customization category (total question 8014), 95 questions from the Data Storage category (total question 6610), 94 questions from Platform Adoption category (total question 5285), 94 questions from Platform Maintenance category (total question 3607), 93 questions from Third-Party Integration category (total question 3247). In summary, we sampled a total of 471 questions.

Step 2. Label Question Types. We analyze and label each question from our samples into the following four categories. The categories and the coding guides follow previous research (Uddin et al. 2021b; Abdellatif et al. 2020; Rosen and Shihab 2016)

- **How-type** post contains a discussion about the implementation details of a technical task (Uddin et al. 2021b). The questions primarily focus on the steps required to solve certain issues or complete certain tasks (e.g., “How to create a submit button template in Oracle APEX?” in [Q1730566](#)).
- **Why-type** post is about troubleshooting and attempting to determine the cause/reason for a behavior. These questions help practitioners understand the problem-solving or debugging approach, e.g., in [Q2517669](#), a user is trying to find out why an SSL server certificate is rejected.
- **What-type** question asks for more information about a particular architecture/event. The practitioners ask for more information that helps them to make informed decisions. For example, in [Q11608661](#) a practitioner is asking for detailed information about the Oracle Apex platform’s secure cookies.
- **Other-type** question do not fall into any of the above three categories, e.g., “Initiating Salesforce API in Google App Script” in [Q66317111](#)

Three authors (first, third and fourth) participated together in the labeling process. We assessed our level of agreement using Cohen kappa(McHugh 2012). The disagreement and annotation difficulties were resolved by discussing with the first author. In general, the authors achieved a substantial agreement ($k > 0.6$) on the 471 questions classified. Our coding guidelines and the final annotated dataset are available in our replication package.

4.3.3 Result

Table 1 shows the percentage of type of questions across our five LCSD topic categories. Similar to related studies (Uddin et al. 2021b; Abdellatif et al. 2020; Rosen and Shihab 2016), during our labeling process, we observed that some of the questions can have multiple labels, e.g., What-type and How-type. For example, “How can I get jQuery post to work with a [platform] WebToLead” in [Q2339550](#) discusses making Ajax request using jQuery

Table 1 Types of questions across the LCSD topic categories

Topic category	How	What	Why	Other
Customization	51.0%	18.4%	17.3%	13.3%
Data Storage	59.8%	13.4%	17.5%	9.3%
Platform Adoption	57.3%	19.8%	9.4%	13.5%
Platform Maintenance	49.0%	20.4%	17.3%	13.3%
Third-Party Integration	61.2%	17.3%	8.2%	13.3%
Overall	55.6%	17.9%	14.0%	12.5%

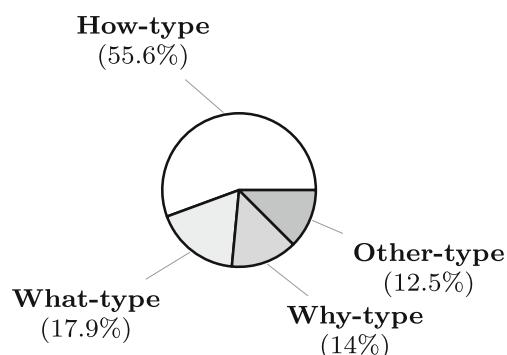
where the practitioner is getting an error response. At the same time, the practitioner is further querying some detailed information on how Ajax requests. Therefore, the sum of percentages of question types reported in the result section is more than 471 classified posts. We now discuss each question type with examples (Fig. 9).

How-type. Around 57% of our annotated questions fall under this type. This type of question is most prevalent in the topic categories Third-Party Integration (61%) followed by Data Storage (60%), Platform Adoption (57%), Customization (51%), Platform Maintenance (49%). This high prevalence is not surprising, given that SO is a Q&A platform and the LCSD practitioners ask many questions about how to implement certain features or debug an issue. Additionally this also signifies that LCSD practitioners are asking a lot of questions while integrating with third-party library (e.g., [Q62825046](#)) and plugins (e.g., [Q61455233](#)) and managing the data with a database management system (e.g., [Q38111768](#)) or file storage (e.g., [Q63284305](#)). To explain further we find questions regarding implementing encryption, e.g., [Q2220076](#), or Making HTTP POST request, e.g., [Q32736416](#), or debugging a script, e.g., [Q45619586](#), or implement a feature, e.g., [Q28990848](#) etc.

What-type. This is the second biggest question type with 18% of all annotated questions. This type of question is the most dominant in the topic categories Platform Maintenance (20.5%), Platform Adoption (20%), and Customization (18%). This type of question can be associated with How-type questions, where the practitioners require further information to implement certain features. For instance, in this question, a practitioner is querying about “How to implement circular cascade select-lists” in [Q60676786](#). The questions in this category signify that practitioners fail to find relevant information from the official documentation (e.g., [Q9377042](#)) sources. Therefore, as this type of question is prevalent in Platform Maintenance and Platform Adoption category, LCSD platform providers might focus more on improving their resources. As an example, we find questions on JavaScript events not working correctly, e.g., [Q51564154](#), roll back changes, e.g., [Q11156810](#), designing workflow, e.g., [Q11156810](#).

Why-type. This is the third most prevalent question type category, with 14% of all annotated questions. This type of question is the most prevalent in the topic categories Customization, Data Storage, and Platform Maintenance with around 17% questions. These questions are mostly related to troubleshooting like when LCSD practitioners implement particular features or deploy an application. For instance, e.g., “Why does this error happen in [Platform]?” in [Q48818859](#), “Why isn’t the document going into edit mode” in [Q51660117](#), “Not able to launch Android Hybrid app using [Platform] Mobile SDK” in [Q20417235](#), “Java code running twice” in [Q17147921](#).

Fig. 9 Distribution of different question Types



Other-type. Around 14% of our annotated questions fall under this type. The questions are almost uniformly distributed across the five topic categories. The questions contain general problems, e.g., “UTF-8 character in attachment name” in [Q22808965](#) or “Domino Server is installed on Unix or Windows?” in [Q10796638](#). Some of the questions in this type also contain multiple/ambiguous questions (e.g., [Q27896327](#)). For example, How to test an application?, which library is better? etc.

RQ3. What types of questions are asked across the observed topic categories? How-type (57%) questions are the most prevalent across all five topic categories, followed by What-type (18%), Why-type (14%), and Other-type (12%) questions. Practitioners in the Customization and Platform Maintenance topic categories are more interested in troubleshooting, i.e. (Why-type, What-type). Practitioners generally ask more implementation questions (i.e., How-type) in the Third-Party Integration Category. Practitioners in the Data Storage topic category are interested in designing databases (i.e., How-type) and troubleshooting (i.e., What-type, Why-type). This indicates the necessity for a more robust community for troubleshooting and debugging issues.

4.4 How are the Observed Topic Categories Discussed Across SDLC Phases? (RQ4)

4.4.1 Motivation

As we observed the prevalence and evolution of diverse LCSD topics in SO, we also find that the topics contain different types of questions. This diversity may indicate that the topics correspond to the different SDLC phases that are used to develop low code software development (see Section 2 for an overview of the LCSD phases). For example, intuitively What-type of questions may be due to the clarification of a low code software requirements during its design phases, which questions/topics related to troubleshooting of issues may be asked during the development, deployment, and maintenance phase. Therefore, an understanding of the SDLC phases in the LCSD questions in SO may offer us an idea about the prevalence of those SDLC phases across our observed LCSD topics in SO. This understanding may help the LCSD practitioners to determine how SO can be used during low code software development across the various SDLC phases.

4.4.2 Approach

In order to understand the distribution of LCSD topics across agile SDLC phases, we collect a statistically significant sample of questions from our extracted dataset D into one of the six Agile software development methodology (Beck et al. 2001) phases: (1) Requirement Analysis & Planning, (2) Application Design, (3) Implementation, (4) Testing, (5) Deployment, and (6) Maintenance. First, we generate a statistically significant sample size. We use the same set of randomly selected (i.e., 471) posts that we produced during RQ3 (see Section 4.3). So, we take a statistically significant stratified random sample for each topic category in our dataset with 95% confidence level and 10 confidence interval to ensure that we have a representative sample from each topic category (Uddin et al. 2021b; Abdellatif et al. 2020). We manually annotate each question post with one/more SDLC phases.

We followed the same annotation strategy to label SDLC phased as we did for RQ3 (see Section 4.3.2). Each question was labeled by at least two authors (second and third/fourth) after extensive group discussion on formalizing annotation guidelines and participating in joint sessions. We find our level of agreement using Cohen kappa (McHugh 2012; Alamin et al. 2021). The authors generally achieved a substantial agreement ($k > 0.70$). For example, a new practitioner is tasked with finding the right LCSD platform during the planning stage of his/her LCSD application. The practitioner queries, “Are there any serious pitfalls to Outsystems Agile Platform?” ([Q3016015](#)). We thus assign the SDLC phase as “Requirement Analysis & Planning”. Another question asks, “Google App Maker app not working after deploy” ([Q42506938](#)). We label the SDLC phase as “Deployment”. For some questions, it involved significant manual assessment to assign appropriate SDLC phase, e.g., Requirement Analysis & Planning phase vs Application Design and Application Design vs Implementation phase. As such, we developed a detailed annotation/coding guide to help us with the manual assessment. This annotation guide was constantly updated during our study to ensure that the guide remained useful with all relevant instructions. For example, one of the questions that helped refine our annotation guide is the question noted by the respected reviewer, i.e., “Can AppMaker be utilized with SQL Server?” in [Q55220499](#). The user in this question wants to know if Google App Maker and SQL Server databases can be connected. This question was categorized as Application design phase. Based on this, according to our annotation guideline, this question can be labelled as Requirement Analysis & Planning phase too. However, after discussion, the first and third authors agreed to label it as Application Design phase because from the problem description, it seems the question mainly focuses on connecting the application to a custom data storage. As this question focuses on data source design, which is often explored during the Application Design phase, we concluded that it should be labeled as such. The labeling of each question to determine the precise SDLC phases was conducted by several co-authors in joint discussion sessions spanning over 80 person-hours.

4.4.3 Results

Figure 10 shows the distribution of our LCSD questions into six agile SDLC phase. We find that the Implementation phase has 65% of our 471 annotated questions, followed by

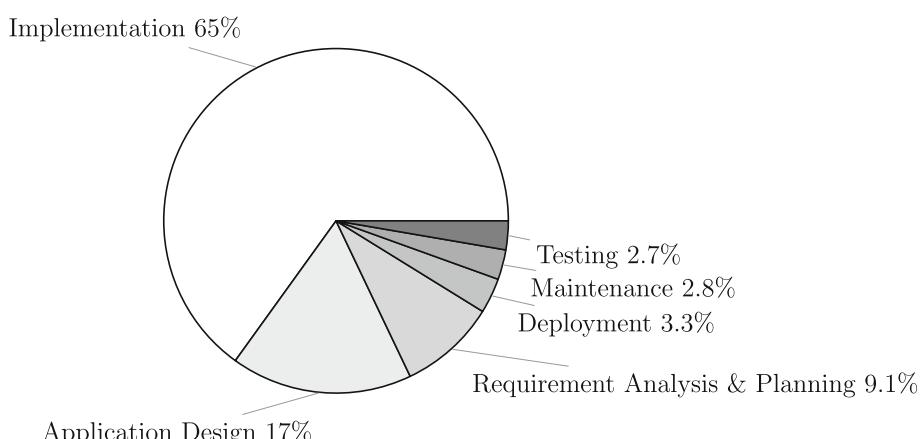


Fig. 10 Distribution of questions (Q) per SDLC phase

Application Design (17%), Requirement Analysis & Planning (9.1%). It is not surprising that the Implementation phase has so many questions because SO is a technical Q&A platform and practitioners use it mostly to find issues when trying to implement some feature. Though the percentage of questions is not very high (e.g., between 2–3%), we also find practitioners ask questions regarding Testing and Deployment phases too (e.g., “Automated Testing for Oracle [Platform] Web Application” in [Q1764497](#)). This analysis highlights that LCSD practitioners ask questions regarding the feasibility analysis of a feature to make design decisions to implement the feature to deployment. We provide an overview of the types of questions asked during these six SDLC phases.

Requirement Analysis & Planning (43, 9.1%) Requirement analysis is the first and most important stage of software development because the application largely depends on this. Requirement analysis is a process to develop software according to the users’ needs. In agile software development methodology, features are implemented incrementally, and requirement and feasibility analysis are crucial in implementing a new feature. During this phase, the operational factors are considered, and the feasibility, time-frame, potential complexity, and reliability. Requirement management tools are typically included with LCSD systems, allowing developers to collect data, modify checklists, and import user stories into sprint plans. Throughout this stage, developers tend to ask questions regarding the platform’s features (e.g., “Does Mendix generates a source code in any particular language, which can be edited and reused?” in [Q53043346](#)), learning curve (e.g., [Q55304547](#), [Q45631057](#)), and the LCSD platform’s support for faster application development (e.g., [Q28983651](#)), general deployment/maintenance support (e.g., [Q50460088](#)) in order to select the best platform for their needs. For example, in this popular question, a new practitioner is asking for some drawbacks on some potential pitfalls for a particular LCSD platform, e.g., “Are there any serious pitfalls to [Platform] Agile Platform?” ([Q3016015](#)). A developer from that platform provider suggests using the platform to build an application and decide for himself as it is hard to define what someone might consider a pitfall. In another question, a practitioner is asking if it is possible to integrate Selenium with an LCSD platform (e.g., [Q52010004](#))

Application Design (80, 17%) The design specification is created in this step based on the application’s needs. The application architecture (e.g., [Q53820097](#)), modularity, and extensibility are all reviewed and approved by all critical stakeholders. The LCSD developers face challenges regarding data storage design, drag and drop UI design, connecting on-premise data-sources with the LCSD platform (e.g., “Can AppMaker be used with SQL Server” ([Q55220499](#))), data migration to LCSD platform ([Q46421271](#)), following best practices (e.g., “Salesforce Best Practice To Minimize Data Storage Size” in [Q14073151](#)), designing a responsive web page (e.g., [Q52744026](#)).

Implementation (306, 65%) The actual application development begins at this phase. LCSD developers confront a variety of obstacles when they try to customize the application (i.e., personalize UI (e.g., [Q6454308](#)), implement business logic (e.g., [Q40472354](#))), integrate third-party plugins (e.g., [Q46538734](#)), debug (e.g., [Q35898112](#)) and test the implemented functionality. For example, LCSD practitioners ask customization questions such as How can they change the timezone in a platform in [Q47731051](#), customizing UI in [Q40159662](#). Many of these challenges arise from incomplete or incorrect documentation. In [Q34510911](#), an LCSD developer asks for sample code to convert a web page to a PDF. The official documentation is not sufficient enough for entry-level practitioners.

Testing (13, 2.7%) LCSD testing differs from standard software testing in some fundamental ways. In LCSD development, many of the features are implemented using graphical interfaces, and they are provided and tested by the LCSD platform providers. As a result, unit testing is less important compared to traditional software development. In LCSD approach practitioners face difficulties to lack of documentation of testing approach in LCSD platform (e.g., “How to bypass login for unit-testing [Platform]?” in [Q54432666](#)), test coverage (e.g., [Q54899980](#), [Q57755398](#)), automated testing (e.g., “[Platform] 20.1 automated testing” [Q63594106](#)), testing browser compatibility (e.g., [Q](#)), troubleshooting errors while running tests (e.g., [Q47254010](#)) etc.

Deployment (16, 3.3%) At this phase, the feature of the application needs to be deployed for the targeted users. One of the goals of LCSD development is to handle many of the complexities of the *deployment and maintenance* phase. Many LCSD platform providers provide advanced Application Life-Cycle Management tools to deploy and maintain the staging (i.e., testing) and the production server (e.g., [Q65124133](#)). However, LCSD practitioners still face many challenges regarding deployment configuration issues ([Q46369742](#)), Domain name configuration (e.g., DNS configuration (e.g., [Q65678735](#)), SSL Configuration (e.g., [Q67186273](#))), accessibility issues such as with public URL ([Q44136328](#), [Q53884162](#)) etc. For example, in this post, a practitioner is having deployment issues (e.g., “[Platform] app not working after deployment” ([Q42506938](#))). A community member provides a detailed description of how to accomplish this in the answer, highlighting the lack of *Official Documentation* for such a critical use-case. There are a few questions concerning delivering an app with a custom URL or domain name (for example, “How to make friendly custom URL for deployed app” in [Q47194231](#)). It was challenging in this scenario because the platform did not have native support.

Maintenance (13, 2.8%) At this phase, the LCSD application is deployed and requires ongoing maintenance. Sometimes new software development life cycle is agile (i.e., incremental) because new issues are reported that were previously undiscovered and request new features from the users. LCSD practitioners face some problems at this phase, such as event monitoring (e.g., [Q64322219](#)), collaboration and developers role management (e.g., “Role based hierarchy in report access” in [Q10436719](#) or [Q52762374](#)), and application reuse (e.g., [Q64276891](#)), application version, i.e., “Do I have the latest version of an [Platform] component?” in [Q45209796](#) or [Q52762374](#), etc.

Topic Categories in different SDLC phases We find that for all five topic categories, LCSD practitioners need some community support from planning to debugging to deployment (e.g., “How does one deploy after building on [platform]” in [Q3952481](#)). We report how LCSD topics and different types of questions are distributed across six

Table 2 Distribution (frequency) of LCSD topics per SDLC phase. Each colored bar denotes a phase (Black = Requirement Analysis, Green = Application Design, Magenta = Implementation, Red = Testing, Blue = Deployment, Orange = Maintenance)

Topics	Development Phases Found in	#Questions
Customization (95)	■ 5 ■ 17	71 1 1 1
Data Storage (95)	■ 6 ■ 16	70 2 1 1
Platform Adoption (94)	■ 17 ■ 21	51 1 1 4
Platform Maintenance (94)	■ 11 ■ 9	46 10 12 6
Third-Party Integration (93)	■ 4 ■ 17	68 1 2 1

Table 3 Types of questions across the Software development life cycle phases

SDLC phase	How	What	Why	Other
Requirement Analysis & Planning(9%)	28%	35%	7%	30%
Application Design(17%)	75%	16%	4%	10%
Implementation(65%)	59%	17%	16%	11%
Testing(3%)	62%	15%	15%	8%
Deployment(3%)	31%	25%	38%	12%
Maintenance(3%)	46%	15%	31%	15%

SDLC phases. Table 2 shows the distribution of SDLC phases for each topic category. Our analysis shows that for the Customization topic Category, most questions are asked during the Implementation (75%) and Design (18%) phases. The most dominant SDLC phase, i.e., the Implementation phase, is most prevalent in Customization (75%), Data Storage (74%), and Third-Party Integration (73%). Requirement Analysis phase is dominant in Platform Adoption (18%) and Platform Maintenance (12%) topic categories where practitioners ask questions like “Disadvantages of the [platform]” in [Q1664503](#). Similarly, question in Platform Maintenance topic category is also prevalent in Testing (11%), deployment (13%), and Maintenance (6%) SDLC stage.

Types of questions in different SDLC phases We report the distribution of question types across SDLC phases in Table 3. It shows that for Requirement Analysis & Planning phase, most questions (35%) belong to What-type. This insight signifies that at this phase, practitioners are making inquiries about feature details (e.g., [Q9577099](#)). In the Application Design, Implementation, and testing phase, most of the questions belong to How-type, i.e., practitioners are querying about how they can implement a particular feature (e.g., [Q13933003](#)) or test it (e.g., [Q9594709](#)). At the Deployment phase most prominent is Why-type (38%) followed by How-type(31%). We can see a similar pattern for the Maintenance phase, where the most significant question type is How-type (46%) followed by Why-type (31%). We see this pattern because, at the Deployment and Maintenance phase, most of the questions belong to some server configuration error (e.g., [Q4497228](#)) and the practitioners’ inquiry about how they can set up specific server settings (e.g., [Q8148247](#)). Similarly, we find that What-type questions are more prevalent during Requirement Analysis and Deployment phases.

RQ4. How are the observed topic categories discussed across SDLC phases? Among six agile SDLC phases, the Implementation phase is the most prevalent (65% questions), followed by Application Design (17%), Requirement Analysis & Planning (9.1%), Deployment (3.3%), Maintenance (2.8%) and Testing (2.7%). The Implementation Phase is most prevalent in all of the five topic categories and four question types. During Requirement Analysis, Testing, and Deployment phases, Platform Adoption and Platform Maintenance topic categories are more dominant. The How-type question is most popular in the Application Design phase, the what-type question is prevalent in the Requirement Analysis and Planning phase, and the why-type question is prevalent in the Deployment and Requirement Analysis phases.

4.5 What LCSD Topics are the Most Difficult to Get an Accepted Answer? (RQ5)

4.5.1 Motivation

After reviewing LCSD-related topics and discussions in the agile SDLC stages, we discovered that LCSD practitioners encounter generic software development problems and particular challenges specific to LCSD platforms (e.g., Platform Adoption, Platform Maintenance). Some posts come up repeatedly, and some have a lot of community participation (i.e., answers, comments, up-votes). As a result, not all topics and SDLC phases are equally difficult to get a solution. A thorough examination of the complexity and popularity of the practitioners' conversation might yield valuable information about how to prioritize research and community support. For example, LCSD platform providers and academics can take the required measures to make the architecture, design, features, and tools of LCSD platforms more useable for practitioners, particularly newbies.

4.5.2 Approach

We compute the difficulty of getting an accepted answer for a group of questions using two metrics for each question in that group (1) Percentage of questions without an accepted answer, (2) Average median time needed to get an accepted answer. In the same way, we use the following three popularity metrics to calculate popularity of that topic in the SO community: (1) Average number of views, (2) Average number of favorites (i.e., for each question number of users marked as favorite), (3) Average score.

The five metrics are standard features of a SO question, and many other related studies (Uddin et al. 2021b; Alamin et al. 2021; Bagherzadeh and Khatchadourian 2019; Abdel-latif et al. 2020; Ahmed and Bagherzadeh 2018) have used them to analyze the popularity and difficulty of getting a solution for a question. In SO, one question can have multiple answers, and The user who posted the question has the option of marking it as accepted. Hence, the accepted answer is considered correct or sound quality. So, the absence of an accepted answer may indicate the user did not find a helpful, appropriate answer. The quality of the question (i.e., problem description) might be one reason for not getting an acceptable answer. However, the SO community collaboratively edits and improves the posts. Therefore, the lack of an accepted answer most likely indicates that the SO community finds those questions challenging to answer. The success and usefulness of a crowd-sourced platform such as SO depends on the community members to quickly provide relevant, helpful correct information. In SO, the median time to get an answer is around 21 min only (Uddin et al. 2021b), but a complicated or domain-specific question may necessitate additional time to receive an accepted answer.

It can be non-trivial to assess the popularity and difficulty of getting an accepted answer for the topics using multiple metrics. We thus compute two fused metrics following related works (Uddin et al. 2021b). We describe the two fused metrics below.

Fused Popularity Metrics First, we compute the popularity metrics for each of the 40 LCSD topics. However, the average view counts can be in the range of hundreds, average scores, and average favorite count between 0-3. Therefore, following related study (Uddin et al. 2021b) we normalize the values of the metrics by dividing the metrics by the average of the metric values of all the groups (e.g., for topics $K = 40$). Thus, we create three new normalized popularity metrics for each topic. For example the normalized metrics for

a group i for all the K groups can be $ViewN_i$, $FavoriteN_i$, $ScoreN_i$ (e.g., for LCSD topics $K = 40$). Finally, We calculate the fused popularity $FusedP_i$ of a group i by taking the average of the three normalized metric values.

$$ViewN_i = \frac{View_i}{\frac{\sum_{j=1}^K View_j}{K}} \quad (5)$$

$$FavoriteN_i = \frac{Favorite_i}{\frac{\sum_{j=1}^K Favorite_j}{K}} \quad (6)$$

$$ScoreN_i = \frac{Score_i}{\frac{\sum_{j=1}^K Score_j}{K}} \quad (7)$$

$$FusedP_i = \frac{ViewN_i + FavoriteN_i + ScoreN_i}{3} \quad (8)$$

Fused Difficulty Metrics Similar to popularity metrics, we first compute the difficulty metrics for each topic. Then we normalize the metric values by dividing them by the average of the metric value across all groups (e.g., 40 for LCSD topics). Thus we, create two new normalized metrics for a given topic i . Finally, We calculate the fused difficulty metric $FusedD_i$ of topic i by taking the average of the normalized metric values.

$$PctQuesWOAccAnsN_i = \frac{PctQuesWOAccAns_i}{\frac{\sum_{j=1}^K PctQuesWOAccAns_j}{K}} \quad (9)$$

$$MedHrsToGetAccAnsN_i = \frac{MedHrsToGetAccAns_i}{\frac{\sum_{j=1}^K MedHrsToGetAccAns_j}{K}} \quad (10)$$

$$FusedD_i = \frac{PctQuesWOAccAnsN_i + MedHrsToGetAccAnsN_i}{2} \quad (11)$$

In addition to this, we also aim to determine the correlation between the difficulty and the popularity of the topics. We use the Kendall Tau correlation measure (Kendall 1938) to find the correlation between topic popularity and topic difficulty. Unlike Mann-Whitney correlation (Kruskal 1957), it is not susceptible to outliers in the data. We can not provide the evolution of popularity and difficulty for these topics because SO does not provide the data across a time series for all metrics such as view count, score, etc. However, asLCSD-related topics are showing increasing trends in recent times, our analysis is valid for recent times.

4.5.3 Results

In Fig. 11 we present an overview of the five high-level topic categories and their popularity and difficulty to get an accepted answer. In the Figure, the bubble size represents the number of questions in that category. The Figure shows that Platform Adoption is the most popular and challenging topic category to get an accepted answer, followed by Customization, Data Storage, Platform Maintenance, and Third-Party Integration. We can also see that three topic categories, Platform Maintenance, Data Storage, and Customization, are almost similar in terms of difficulty to get a solution. From our analysis, we find that practitioners find the Third-Party Integration topic category relatively less difficult because many questions in this category are also relevant to traditional software development (e.g., integrating Google Maps in [Q63457325](#) and [Q1258834](#)) and thus easier to get community support. Similarly, we

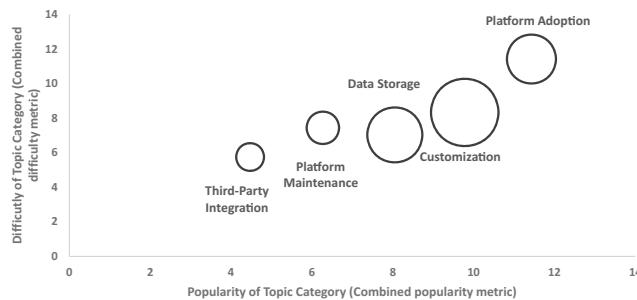


Fig. 11 The popularity vs. difficulty of getting an accepted answer for LCSD Topic categories

find that questions in the Platform Adoption topic category are quite specific to particular LCSD platforms and thus sometimes have less community support to find an acceptable answer quickly.

Topic Popularity For each of the 40 topics, Table 4 shows three popularity metrics: Average number of 1. Views, 2. Favorites, 3. Scores. It also contains the combined popularity metrics (i.e., FusedP) that are based on the above three metrics and using the (8). In the Table, the topics are presented in descending order based on the FusedP popularity metric.

Platform Related Query topic from the Platform Adoption Category has the highest FusedP score. It also has the highest average favorite count (e.g., 0.90) and highest average score (e.g., 2.60). 1.7% of total questions. This topic contains discussion about LCSD platforms features of different platforms, software development methodologies such as Agile and RAD development. The topic Message Queue under Platform Adoption category has the second highest FusedP value. This topic is about different asynchronous service-to-service data exchange mechanisms such as using a message queue. It generally contains discussions about popular micro-service design patterns. The topic Dynamic Page Layout under Customization categories is the third most popular topic and it has the highest average view count (e.g., 2447.2). The posts under this topic discuss about UI (i.e. page) customization, hiding or moving elements based on some user action or an event (e.g., disable a button for dynamic action in [Q8640964](#)). The eSignature topic from Third-Party Integration is the least popular with only 1.15% of total questions, a fused value of 0.52. It has the lowest favorite and score count. This contains discussion about different issues and customization for electronic signature of documents, i.e., docusign about collecting user's agreement/permission for sales or account opening. This topic is not that much popular and easy to get an accepted answer because this requirement is not generalized and not all the low-code application requires this.

Topic Difficulty In Table 5 we present the two difficulty metrics: for all the questions in a topic 1. Percentage of questions without accepted answers, 2. Median hours to get accepted answer. Similar to topic popularity, we also report the combined topic difficulty metrics (e.g., FusedD) using the (11) and the above two difficulty metrics. The topics in Table 5 are presented in descending order based on the FusedD value.

Topic Message Queue under Platform Adoption category is the most difficult topic to get an accepted answer in terms of FusedD value. Most median hours to get accepted answers (21). This topic contains discussion about general micro-service architecture (i.e., producer and consumer) and well as LCSD platform-specific support for these architectures. This

Table 4 Popularity for getting an accepted answer for LCSD topics

Topic	Category	FusedP	#View	#Favorite	#Score
Platform Related Query	Platform Adoption	3.45	2229.9	0.9	2.6
Message Queue	Platform Adoption	1.49	1671.7	0.3	1.1
Dynamic Page Layout	Customization	1.31	2447.2	0.2	0.7
Build Config. Management	Platform Maintenance	1.22	1522.1	0.2	1
Pattern Matching	Customization	1.17	1539.5	0.2	0.9
SQL CRUD	Data Storage	1.15	1615.7	0.2	0.8
Web-Service Communication	Platform Adoption	1.15	1454.9	0.2	0.9
Misc. SWE Discussion	Platform Adoption	1.13	1193.4	0.2	1
Interactive Report	Platform Adoption	1.12	1680.8	0.2	0.7
Fetch & Process API Response	Third-Party Integration	1.11	1277.8	0.2	0.9
Data Security & Replication	Data Storage	1.1	1411.7	0.2	0.8
Hosting Config. & SEO	Platform Maintenance	1.09	1377.9	0.2	0.8
Asynchronous Batch Jobs	Platform Maintenance	1.06	1245.7	0.2	0.8
Authentication & Authorization	Platform Adoption	1.05	1057.6	0.2	0.9
Library Dependency Mngmt	Platform Maintenance	1.05	1230.6	0.2	0.8
Email Processing	Third-Party Integration	1.05	1600	0.2	0.6
Formatted Data Parsing	Customization	1	1607.8	0.1	0.9
File Management	Data Storage	0.98	1302.6	0.2	0.6
DB Setup & Migration	Data Storage	0.97	1282.9	0.2	0.6
Testing	Platform Maintenance	0.96	1810.5	0.1	0.7
Date & Time Manipulation	Customization	0.94	1720.5	0.1	0.7
DB Stored Procedure	Data Storage	0.94	1715.5	0.1	0.7
Dynamic Data Binding	Customization	0.92	1810.5	0.1	0.6
External Web Req Processing	Third-Party Integration	0.91	831.8	0.2	0.7
App Deployment	Platform Maintenance	0.89	757.3	0.2	0.7
Semi-Structured Data Proc.	Data Storage	0.88	1096.5	0.2	0.5
Upgradation & Compatibility	Third-Party Integration	0.88	559.1	0.2	0.8
Dialog Box Manipulation	Customization	0.79	1479.4	0.1	0.5
Dynamic Event Handling	Customization	0.77	1245.1	0.1	0.6
Dynamic Form Controller	Customization	0.76	1382.1	0.1	0.5
Conditional BPMN	Customization	0.75	1335.1	0.1	0.5
SQL Sysntax Error	Data Storage	0.75	1343	0.1	0.5
Graph/Chart	Platform Adoption	0.75	1156.3	0.1	0.6
User Role Management	Platform Adoption	0.73	1058.3	0.1	0.6
Window Style Manipulation	Customization	0.71	1000	0.1	0.6
Entity Relationship Mgmt	Data Storage	0.71	1178	0.1	0.5
Dynamic Data Filtering	Customization	0.66	1148.1	0.1	0.4
Date-based filtering	Data Storage	0.57	781.9	0.1	0.4
Platform Infrastructure API	Platform Adoption	0.56	577.2	0.1	0.5
eSignature	Third-Party Integration	0.52	574.9	0.1	0.4

is why this topic is also second most popular topic. Library Dependency Mngmt topic from Platform Maintenance is the second most difficult topic to get an accepted answer. Around 70% of its questions do not have any accepted answers. This topic concerns differ-

Table 5 Difficulty for getting an accepted answer for LCSD topics

Topic	Category	FusedD	Med. Hrs To Acc.	Ques. W/O Acc.
Message Queue	Platform Adoption	1.86	21.4	61
Library Dependency Mngmt	Platform Maintenance	1.78	18.8	70
Web-Service Communication	Platform Adoption	1.76	19.8	60
Authentication & Authorization	Platform Adoption	1.67	17.4	68
Platform Infrastructure API	Platform Adoption	1.62	16.3	70
Fetch & Process API Response	Third-Party Integration	1.6	16.8	64
External Web Req Processing	Third-Party Integration	1.37	13.1	64
App Deployment	Platform Maintenance	1.35	12.1	70
Hosting Config. & SEO	Platform Maintenance	1.35	12.6	65
eSignature	Third-Party Integration	1.32	11.4	71
File Management	Data Storage	1.24	11.4	62
Asynchronous Batch Jobs	Platform Maintenance	1.19	10.8	60
Dynamic Page Layout	Customization	1.15	10.1	61
User Role Management	Platform Adoption	1.03	8.3	60
Graph/Chart	Platform Adoption	0.99	6.6	68
Platform Related Query	Platform Adoption	0.99	8.4	54
DB Stored Procedure	Data Storage	0.97	7.7	57
DB Setup & Migration	Data Storage	0.95	6.9	60
Dynamic Form Controller	Customization	0.92	6.3	61
Testing	Platform Maintenance	0.92	6.7	59
Conditional BPMN	Customization	0.85	5.7	58
Build Config. Management	Platform Maintenance	0.85	6.4	53
Dynamic Event Handling	Customization	0.84	4.2	68
Semi-Structured Data Proc.	Data Storage	0.82	4.6	63
Email Processing	Third-Party Integration	0.8	4.7	59
Dialog Box Manipulation	Customization	0.79	4.8	57
Interactive Report	Platform Adoption	0.79	5	56
Dynamic Data Binding	Customization	0.77	5.1	53
Dynamic Data Filtering	Customization	0.71	4.8	48
Misc. SWE Discussion	Platform Adoption	0.71	4.8	48
Data Security & Replication	Data Storage	0.66	3.5	52
Upgradation & Compatibility	Third-Party Integration	0.66	4.1	48
Date-based filtering	Data Storage	0.65	2	61
SQL Syntax Error	Data Storage	0.62	1.7	60
Entity Relationship Mgmt	Data Storage	0.62	2.2	57
Formatted Data Parsing	Customization	0.61	3.1	49
Date & Time Manipulation	Customization	0.59	2.5	51
Window Style Manipulation	Customization	0.58	2.3	51
Pattern Matching	Customization	0.52	1.8	48
SQL CRUD	Data Storage	0.50	1.7	46

ent troubleshooting issues about library and dependencies of the system, server configuration, different library version compatibility issues. Web-Service Communication topic from Platform Adoption is the third most difficult topic. It has a long median wait time (around 20

Table 6 Correlation between the topic popularity and difficulty

Coefficient/p-value	View	Favorites	Score
% Ques. w/o acc. ans	-0.33/0.01	0.02/0.88	-0.17/0.15
Med. Hrs to acc. ans	-0.05/0.65	0.30/0.02	0.22/0.05

hours) to get an accepted answer. This topic contains discussions about service-to-service communication via web service description language, HTTP REST message, and Windows Communication Foundation.

The topics that contain discussion about general software development (not specific to LCSD platforms) are the least difficult topics to get an accepted answer. For example, topic SQL CRUD under Data Storage category is the least difficult topic in terms of FusedD value (e.g., 0.5). This contains database CRUD related queries, and advanced queries too, such as inner join, nested join, aggregate. This also contains discussion about Object query language, which is a high-level wrapper over SQL. Topic SQL CRUD and SQL Syntax Error from the Data Storage category are two of the least difficult topics in terms of median hours to get accepted answers. Topic Pattern Matching and SQL CRUD are two of the least difficult topics in terms of questions without accepted answers.

Alternatively, topics that are specific to LCSD platforms are the most difficult topics. Four out of five most difficult topic belongs to Platform Adoption Categories. These questions can be popular as well as difficult. For example, LCSD-related Third-Party Integration related topic eSignature is the least popular topic from Table 4, is the most difficult topic in terms of questions without accepted answers (71%). Topic Platform Related Query is in the mid-range in terms of difficulty but most popular to get an accepted answer.

Correlation Between Topic Difficulty and Popularity Here we want to explore if there is any positive or negative relationship between topic popularity and difficulty. For example, Message Queue is the most difficult and, at the same time second most popular topic to get an accepted answer in terms of FusedD and FusedP metrics. Platform Related Query is the most popular but mid-range difficult topic.

Table 6 shows six correlation measures between topic difficulty and popularity in Tables 4 and 5. Three out of six correlation coefficients are negative, and the other three are positive and they are not statistically significant with a 95% confidence level. Therefore, we can not say the most popular topic is the least difficult to get an accepted answer and vice versa. Nonetheless, LCSD platform providers could use this insight to take necessary steps. Most popular topics should have an easy to access-able answer (i.e., least difficult).

RQ5. What LCSD topics are the most difficult to answer? Platform Adoption is the most popular and challenging topic category, followed by Customization, Data Storage, Platform Maintenance, and Third-Party Integration. We also find that LCSD practitioners find Software Deployment and Maintenance phase most popular and difficult and Testing phase to be least difficult to an accepted answer. This indicates that LCSD platform providers should provide additional support to enable low-code practitioners understand and utilize the platform's features.

5 Discussions

During our analysis, we observed that several LCSD platforms are more popular across the topics than other platforms. We analyze our findings of LCSD topics across the top 10 most prevalent LCSD platforms in the dataset (Section 5.4). Finally, we discuss the implications of our study findings in Section 5.6.

5.1 Issues with Not Accepted Answers or Posts with Negative Score

In this paper, for topic modeling we used questions and accepted answers only. We did not consider the posts with negative score too because of the following observations. (1) Many other similar empirical studies on Topic modeling on SO posts also considered the questions and accepted answers only, e.g., IoT developers discussions in SO (Uddin et al. 2021b), big data related discussion (Bagherzadeh and Khatchadourian 2019), concurrency related topics (Ahmed and Bagherzadeh 2018), mobile app development (Rosen and Shihab 2015). (2) A significant number of studies (Asaduzzaman et al. 2013; Ren et al. 2019; Ponzanelli et al. 2014; Yang et al. 2016a) report quality of questions and unaccepted answers in SO are questionable and therefore it is quite a standard practice for SE researchers to consider the accepted answers in SO only. For example, In [Q7504057](#) (Fig. 12) a user asks question about python code/package to connect and retrieve data from Salesforce. The accepted answer [A7504244](#) provides a relevant python code snippet the unaccepted answer [A34055640](#) provide resource link for a command line tool which may be relevant but exactly not what the user asked for. (3) Negative scored questions are typically incorrectly tagged (e.g., [Q4862071](#), [Q21377026](#), [Q37371712](#)), duplicates (e.g., [Q12282151](#), [Q48121405](#)), lack a detailed problem description (e.g., [Q25691340](#), [Q1974480](#), [Q50666660](#)), lack correct formatting (e.g., [Q32208310](#)). For instance, in [Q51635004](#) (Fig. 12) a user inquires about an error encountered when attempting to contact a Zoho API. However, crucial important information such as an issue code or error message is lacking from the question description. In [Q4862071](#), an inexperienced user inadvertently tagged a question about the Oracle Apex platform with the Salesforce tag. We, therefore, choose not to include questions with a negative score or unaccepted answers. We also provide potentially missing out some insights for this choice in the threats to validity section (Section 6).

5.2 Discontinued Low-Code Platforms and Future Trends

From our analysis on Section 4.2 we see the evolution of LCSD platforms, especially from 2012. According to our data, we can see the discontinuation of some low-code platforms but they are usually soon replaced by new low-code/no-code services. For example, In Jan 2020, Google announced the discontinuation of Google App Maker (2020) by 2021 (2021). But, shortly thereafter, Google announced a “no-code” platform called “AppSheet” (2021) and promoted their fully managed serverless platform called AppEngine (2021) to create web application promoting low-code approach. Microsoft and Amazon are also competing for superior low-code/no-code platforms with the emergence of new low-code service platforms such as Microsoft Power FX (2021), Amazon Honeycode (2022), AWS Amplify Studio (2022). The low-code approach is attracting increasing interest from traditional businesses, particularly during the pandemic (2022).

Create Zoho campaign with api

Asked 4 years ago Modified 4 years ago Viewed 192 times

I tried to create a zoho campaign following this link
<https://www.zoho.com/campaigns/help/api/create-campaign.html>

-4
but every time, I get an error.

This is the link that I used

https://campaigns.zoho.com/api/v2/createmain?campaignName=mnewone&from_email=sammoudi.maher@gmail.com&subject=tesSubject&authtoken=XXXXXXXXXXXXXX&refmt=json

Where is the issue?

python salesforce library to get salesforce data?

Asked 10 years, 11 months ago Modified 11 months ago Viewed 14k times

Share Edit Follow Flag

asked Sep 21, 2011 at 17:32

daydreamer 81.0k ● 577 ● 431 ● 606

Although this is not Python specific, I came across a cool tool for the command line. You could run bash commands as an option.

0 https://forcecli.herokuapp.com/

User: force <command> [<args>]

Available commands:
login Log in to the force.com instance
logout Log out from force.com
help List available commands
active Show or set the active force.com account
update Update the active force.com account
describe Describe the object or list of available objects
soql Execute a SOQL statement
soqlCustom Execute a SOQL statement against custom objects
bigobject Manage big objects
field Create, update, or delete fields
record Create, modify, or view records
bulk Create, update, or delete records
fetch Export specified artifact(s) to a local directory
import Import metadata or artifacts to a local directory
execute Execute a SOQL statement
query Execute a SOQL statement
queryMore Continue a previous SOQL query
trace Manage trace flags
log Manage log files
event Logfile, list and fetch event log file
search Search for records using keywords
test Run apex tests
security Display security rules, DLS and PLS for a give SOObject
version Display current version
upgrade Upgrade the org to the latest version
push Deploy artifact from a local directory
pull Pull artifact from a remote location
password See password status or reset password
notifier Manage notifications
limits Display current limits
help Help information
dataspace Manage DataPipes

Share Edit Follow Flag

answered Dec 3, 2015 at 0:20

user4199456

(a) A question with negative score

(b) A question with irreverent unaccepted answer

Fig. 12 SO questions with a negative score or unaccepted SO answer

5.3 LDA Parameter Analysis

In this study, we applied LDA topic modelling, which employs Dirichlet distribution, to identify practitioners' discussions on low-code. As described in details in Section 3.2, we followed the industry standard to configure the parameters and hyperparameters and also followed the industry recommendation to manually annotate the topics as described in Section 4.1 in order to avoid sub-optimal solutions (De Lucia et al. 2014). Following similar studies (Uddin et al. 2021b; Abdellatif et al. 2020; Han et al. 2020) we use the coherence score of each model for different values of K . However, since LDA itself is probabilistic in nature (Agrawal et al. 2018) and can produce different results different runs on the same low-code dataset. In order to mitigate this problem, we run our LDA model

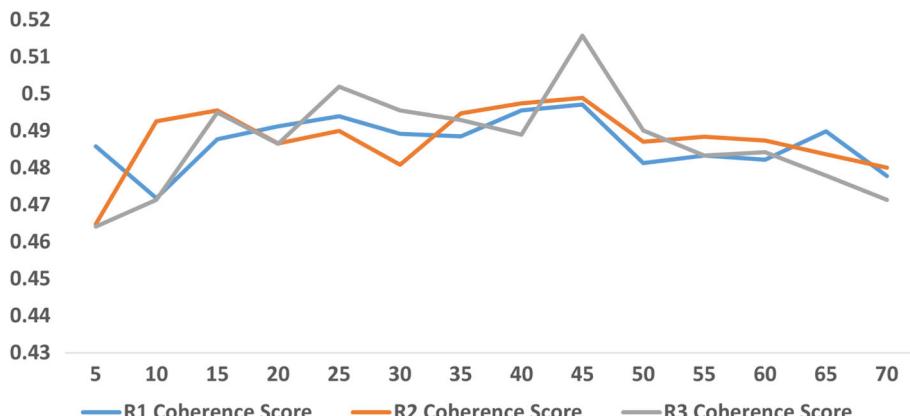


Fig. 13 The different coherence values for varied K on different runs

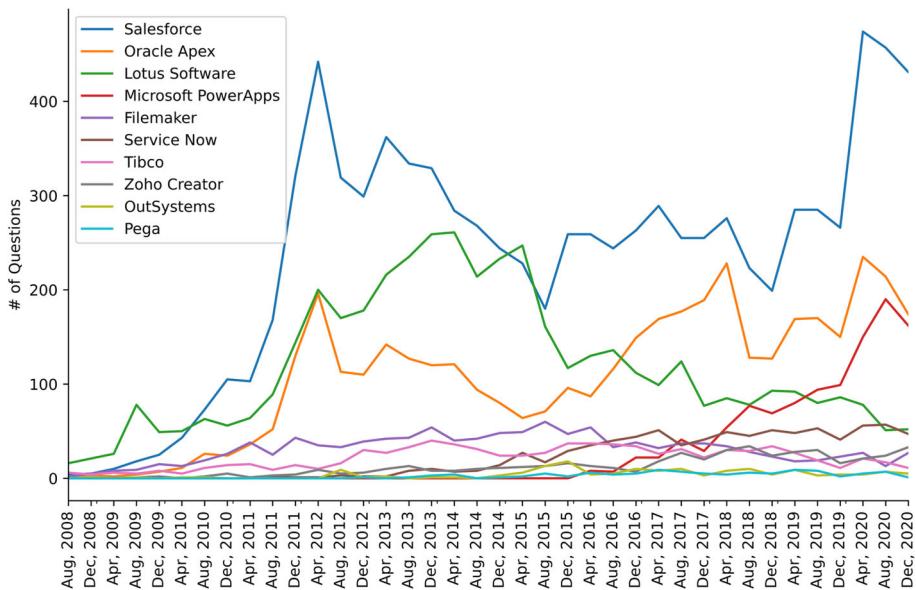


Fig. 14 The evolution of top ten LCSD platforms discussions over time

three times and compare the optimal number of topics. Figure 13 shows the result of different coherence score for different values of K . Moreover, we can see after reaching highest coherence values for $K = 45$ the overall coherence score decreases as the value of K increases.

5.4 The Prevalence & Evolution of Top Ten LCSD Platforms

Our analysis of the evolution of topic categories (see Section 4.2) shows that there is an overall increase in the number of new questions across the topics in SO. Our SO dataset is created by taking into account the LCSD platforms. In Fig. 14, we show how the 10 LCSD platforms evolve in our SO dataset over the past decade based on the number of new questions. Salesforce (2022) is the biggest and one of the oldest LCSD platforms (released in 1999) in our dataset with around 30% of all questions followed by Lotus Software (2022), Oracle Apex (2022), Microsoft powerapps(Microsoft Power Apps Platform Overview 2022). Among these platforms, IBM Lotus Software was quite popular during the 2014s and gradually lost its popularity, and IBM finally sold it in 2018. Salesforce platform has been the most popular platform in terms of SO discussions since 2012. Our graph shows that these other three platforms, especially Microsoft Powerapps, are gaining lots of attention during the pandemic, i.e., early 2020.

We provide more context for these platforms in Fig. 15 by illustrating the distribution of our observed five topic categories across the top ten LCSD platforms. We can see that Powerapps have the most number of queries in the Customization and Platform Adoption category. This happens because Powerapps is a relatively new LCSD platform (released in 2016) and it is gaining more and more attention from the community, and thus there are more queries such as business logic implementation (i.e., [Q61685582](#)), connect Powerapps to a database in [Q61611950](#), user permission (i.e., [Q61838119](#)). We can also see that older

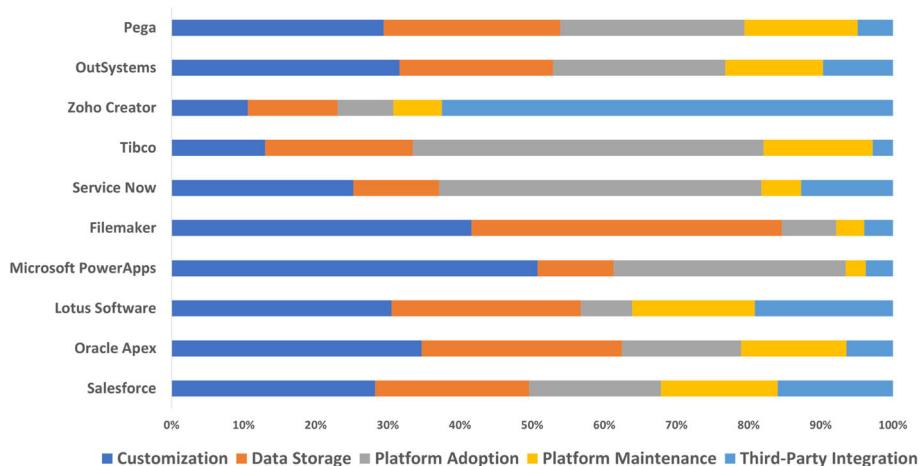


Fig. 15 The distribution of topic categories across top ten LCSD platforms

platforms such as Salesforce and Oracle Apex have more queries regarding Platform Maintenance, Third-Party Integration topic category. Practitioners ask many different questions regarding these platforms such as deployment-related (e.g., “Deploying a salesforce.com flex app without visualforce” in [Q6614226](#)), third-party API integration (e.g., “Google Map Integrated with Salesforce is Blank” in [Q9028682](#)), maintenance deployment (e.g., “Salesforce deployment error because of test class failure” in [Q9171945](#)), interactive report in [Q9700660](#), customization with JSON [Q9833992](#), “what is dashboard?” in [Q10911269](#) and Oracle Apex how to use platform in [Q9438695](#). Platform Adoption is a prevalent topic category in the Powerapps, ServiceNow, and Tibco platforms. We also notice that the Data Storage category is quite popular in Filemaker and Lotus Software. Interestingly we see that Zoho Creator LCSD platform around 60% questions belong to Third-party API integration (especially email configuration [Q48865565](#)). This data sheds light on the popular discussion topics of more recent and earlier LCSD platforms.

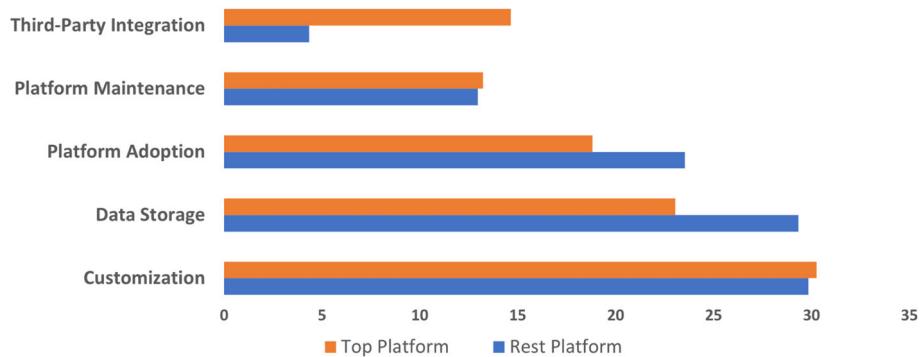


Fig. 16 The percentage of questions distributed across five topic categories for the top ten LCSD platforms versus the rest of the platforms

5.5 The Case of “Aggregating” Data Across Multiple LCSD Platforms

In this study, we analysed 38 LCSD platforms and these platforms have distinct characteristics and challenges. Our goal is to offer researchers and practitioners a comprehensive overview of the LCSD domain as a whole, as opposed to focusing on a single LCSD

Table 7 The summary of our core findings and recommendations

RQ	Theme	Core findings	Recommendations
1	Platform Customization	We identified 40 LCSD topics that fall into five categories. Customization of platform features continues to be the most discussed challenge, referring to the difficulty developers face when seeking to adapt a given feature to a particular situation.	The difficulty with customization can be observed when the LCSD platforms do not offer anyway to change the default interface/functionality. The platforms may offer coding interface where individual components (e.g., a button) can be customized by writing code.
	API Integration	Around 12% of the topics discuss challenges related to the integration of third party apps into an LCSD platform like the integration of REST services.	The LCSD platforms may introduce specialized and programmable interfaces to support the call of a REST service and to be able to process the service output into a format that can be usable within the platform.
2	Platform Adoption	Topics related to customization and platform adoption are being discussed in recent years, because developers are increasingly asking for more documentation and other supporting features as the platforms are being adopted in real-world scenarios.	While the official documentation resources can be further improved by the LCSD vendor, automated tools can be investigated to generate documentation by learning of existing adoption of the platforms.
3	How to Use	More than 50% of the questions are How-type, showing the needs for better learning resources.	The official documentation of implementation can be further enhanced by processing the crowd-shared knowledge of the usage discussions of the platforms (e.g., from Stack Overflow)
4	Server Setup	What and Why-type questions are most dominant for server setup related.	LCSD platform should provide better visualizer and debugger for the practitioners to improve troubleshooting.
	Development Effort	Implementation is the most dominant SDLC phase (65%). Interestingly we also around one third of the are related to Application Design (17%) and Requirement Analysis & Planning (9%)	LCSD platform providers should take necessary steps to provide better community support in SO to address these challenges. Practitioners should also be aware LCSD platforms can improve the development of traditional software development team but they are not yet panacea for all problems.

Table 7 (continued)

RQ	Theme	Core findings	Recommendations
5	Deployment vs Maintenance	Questions related to both the deployment-SDLC and Maintenance-SDLC as the most popular and hardest to get accepted answers.	Platform providers should provide better level of abstraction for cloud management, application deployment and monitoring. Educators can provide better resources to learn about cloud platforms.
	Messaging	The topic message queue from platform adoption is the most difficult to get an accepted answer. This topic contains discussion about the adoption of a general micro-service architecture within LCSD platforms	LCSD platforms can improve the adoption of micro-service architecture and the communication between different the microservices with better message queuing

platform. Hence, we integrated the data from all of these platforms. For instance, Fig. 14 demonstrates that some of the most popular platforms such as Salesforce, Oracle Apex, and Microsoft Powerapps have more questions in SO than other LCSD platforms. Figure 15 demonstrates that questions across these platforms over different topic categories differs slightly. However, Fig. 16 shows that questions for Application Customization and Platform Maintenance topic category for top ten platforms vs others remain about the same at around 30% and 13% respectively. Popular platforms have more questions related to Third Party API integration (15%) than others (4%). The top ten platforms have relatively fewer questions (15%) in Data Storage (23% vs 29%) and Platform Adoption (19% vs 24%) Category compared to other platforms. Overall, we found that the observed topics are found across all the platforms that we studied. Given the popularity of some platforms over others, it is understandable that those platforms are discussed are more and as such some platforms can have more coverage (in terms of number of questions) in a topic over other platforms. However, the prevalence of all platforms across each topic shows that the topics are generally well-represented across the platforms.

5.6 Implications

In Table 7, we summarize the core findings of our study and provide recommendations for each findings. The findings from our study can guide the following three stakeholders: (1) LCSD platform Providers to improve the documentation, deployment, and maintenance support, (2) LCSD Practitioners/Developers to gain a better understanding of the trade-offs between rapid development and customization constraints, (3) Community of LCSD Researchers & Educators to have a deeper understanding of the significant challenges facing the broader research area to make software development more accessible. We discuss the implications below.

In this empirical study, we infer implications and recommendations based on our observation of practitioners' discussion in SO. So further validation from the developers' survey can provide more insight. However, the diversity of the low-code platforms and topics makes

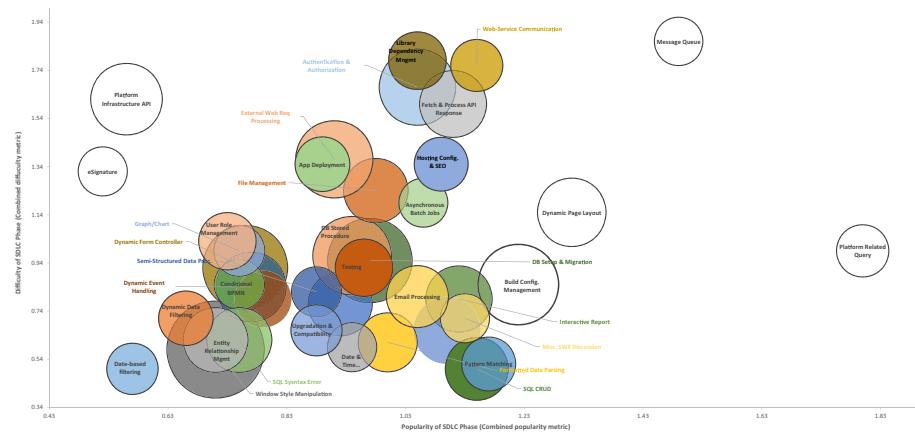


Fig. 17 The popularity vs. difficulty bubble chart for 40 LCSD-related topics

it non-trivial to design a proper survey with representative sample of LCSD practitioners. Therefore, the findings can be used to design multiple LCSD related surveys focusing on different low-code topics and platforms (Fig. 17).

LCSD Platform Vendors. In order to better understand the issues of LCSD, we present a bubble chart with difficulty and popularity of different aspects of LCSD such as Topic Category in Fig. 11, Types of questions in Fig. 19 and agile SDLC phases in Fig. 18. These findings coupled with the evolution of LCSD platforms (14) and discussions (7) shows that Customization and Data Storage related queries are more prevalent, with the majority of these queries occurring during Implementation agile SDLC stage.

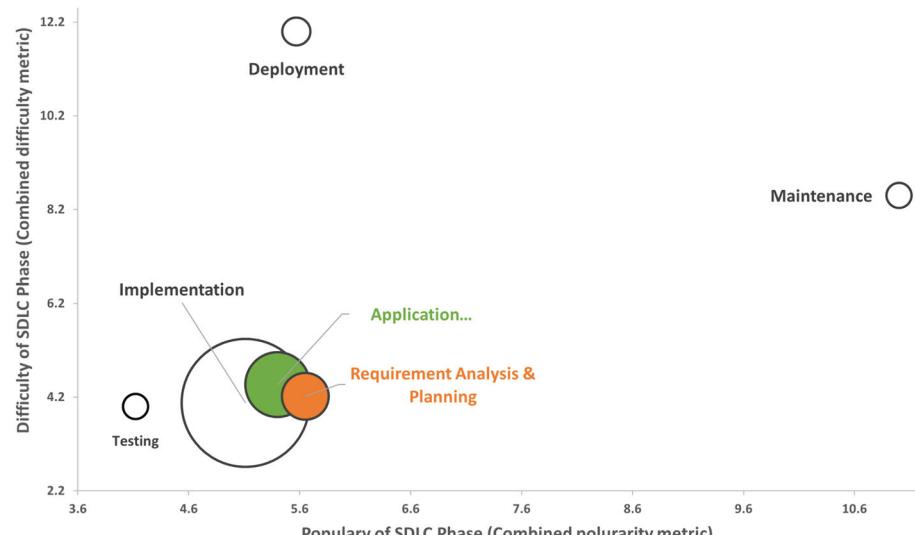


Fig. 18 The popularity vs. difficulty bubble chart for low-code software development life cycle phases

However, one of our interesting findings is *Platform Adoption* related queries are increasing in popularity. LCSD practitioners find LCSD platform infrastructure and server configuration-related quires tough and popular during the Deployment and Maintenance phase. The top five most challenging topics belong to Platform Adoption and Maintenance topic category.

Many new practitioners make queries regarding LCSD platforms, learning resources, basic application and UI customization, and how to get started with this new emerging technology. Figure 17 shows that *Platform Related Query* topic is the most popular among LCSD practitioners. We find that *Documentation* related queries are both top-rated and challenging. Our findings also suggest that many practitioners still face challenges during testing, especially with third-party testing tools like JUnit (in [Q9811992](#)) and troubleshooting. Consequently, many of the questions on this topic remain unanswered. It reveals that to ensure smooth adoption of the LCSD platforms, providers should provide better and more effective documentation and learning resources to reduce entry-level barriers and smooth out the learning curve.

LCSD Practitioners/Developers. Gartner ([2022](#)) estimates that by 2022, more than half of the organizations will adapt LCSD to some extent. Additionally, our analysis reveals a rising trend for LCSD approaches, particularly during Covid-19 pandemic (Fig. 6). We can also see that new LCSD platforms such as Microsoft Powerapps are gaining many developers' attention. LCSD platform enables practitioners with diverse experience to contribute to the development process even without a software development background. However, our finding shows that practitioners find debugging, application accessibility, and documentation challenges. Hence, the practitioners should take the necessary steps to understand the tradeoffs of LCSD platforms' features deeply. The project manager should adopt specific strategies to customize, debug, and test the application. For example, many practitioners struggle with general Third-Party API integration and database design and query. We find that DevOps-related tasks such as CI/CD, Server configuration, and monitoring-related queries are most challenging to the practitioners. So, a well-functioning LCSD team should allocate time and resources to them. It provides valuable insights for project managers to manage resources better (i.e., human resources and development time).

Figure 18 shows that Maintenance is the most popular development phase, followed by Deployment, and Testing is the least popular SDLC phase. Similarly, the figure also shows that questions asked in deployment phase are the most difficult followed by Maintenance. Implementation, Requirement analysis and planning, Application design phase are in the middle range in terms of popularity and difficulty spectrum. Thus, our analysis indicates that LCSD practitioners face more broad and complex application maintenance and deployment-related challenges, on which LCSD platform vendors should concentrate their efforts. This finding can influence the decision-making process of LCSD developers and practitioners like prioritizing their efforts during the design, development, and deployment of software that uses LCSD platforms. For example, if sufficient support or tools are not available for scalable usage and deployment of an LCSD platform, developers may look for alternatives that have better deployment and maintenance support.

One fundamental shortcoming of LCSD platforms is that their abstraction and feature limitations can make customization and debugging extremely difficult. Additionally, managed cloud platforms make data management and deployability more challenging (Sahay et al. [2020](#); Luo et al. [2021](#)). The findings in this study help to present some

strengths and limitations of the overall LCSD paradigm, which complements the findings of other studies (Sahay et al. 2020; Alsaadi et al. 2021; Luo et al. 2021; Waszkowski 2019; Adrian et al. 2020; Ness and Hansen 2019). The analysis could assist LCSD teams in selecting the appropriate LCSD platforms, which is critical for future success.

LCSD Researchers & Educators. The findings of this study have many implications for researchers and educators of LCSD platforms and the border research community to improve the software development process. We discover that What-type and How-type questions are popular among LCSD practitioners. They also find them challenging because of adequate usable documentation. Thus, practitioners ask questions about certain limits or how to implement certain features, and in the accepted answer, some other user simply points to the official documentation page (e.g., “Domino Data Service API Documentation” in [Q59739877](#) and [Q5806293](#)). Many of the challenges faced by low-code petitioners are similar to traditional software developers. So, researchers from border software engineering domain can contribute to improving aspects such as improving documentation (Bayer and Muthig 2006; Khan et al. 2021a; Bhat et al. 2006), improving API description usage (Uddin and Robillard 2015a; Uddin et al. 2021a) and make it more accessible to general practitioners. In the Customization and Data Storage topic category, we find practitioners asking help in generic programming queries, database design, and file management. So, research on those topics will also help the adoption of LCSD. Some LCSD platforms provide great in-build support for unit and functional testing. However, we find around 2.1% of questions belong to Testing topic. Most of these LCSD platforms heavily rely on cloud computing, and thus research improvement of server configuration and library management, i.e., DevOps (Zhu et al. 2016) in general, will aid in better platforms. On the other hand, educators can focus their efforts on making the learning resources on Automatic testing, Server config. and DevOps practices such as CI/CD more accessible to the citizen developers.

Figure 19 shows What-type of posts are most popular, followed by Why-type, How-type, and Others-type. Additionally, it demonstrates that the most challenging question type is Why-type, followed by What-type, How-type, and others. So, although How-type questions are most dominant, the details types (i.e., Why-type, What-type) of questions are more popular and challenging. This analysis implies that LCSD practitioners have a harder time

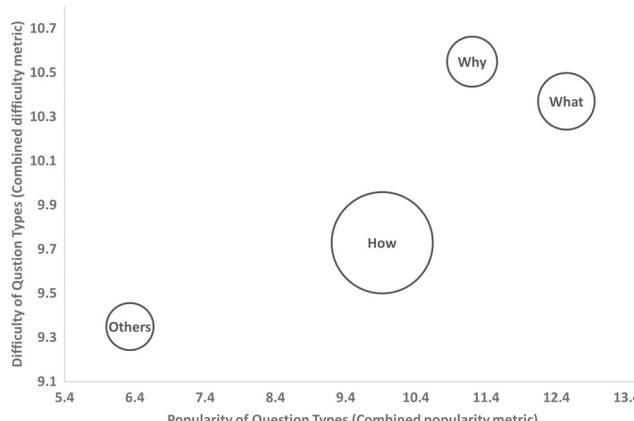


Fig. 19 The popularity vs. difficulty of different types of LCSD-related questions

finding detailed information regarding different platform features. As a result, LCSD platform providers should improve their documentation. Intuitively, How-type questions can be answered with better documentation for LCSD platforms. Given the official API documentation can often be incomplete and obsolete (Uddin and Robillard 2015b; Khan et al. 2021b) and given that our research shows that LCSD developers use SO to ask questions about various topics, LCSD researchers can develop techniques and tools to automatically improve the documentation of LCSD platforms by analyzing SO questions and answers. Indeed, existing research efforts show that solutions posted in SO can be used to improve API documentation by supporting diverse development tasks and programming languages (Chakraborty et al. 2021; Uddin et al. 2019, 2020a, b; Uddin and Khomh 2017b, c, 2019).

We find that the LCSD paradigm's challenges can be different from traditional software development (Sahay et al. 2020). Simultaneously, researchers can study how to provide better tools for practitioners to customize the application. Security is an open research opportunity for such platforms as a security vulnerability in such platforms or frameworks could compromise millions of applications and users (Lin et al. 2020). Researchers can develop better testing approaches to ensure faster development and dependability. Educators can also benefit from the results presented in Table 4 and Fig. 17 to prioritize their focus on different topics such as *Library Dependency Mngmt*, *Web-Service Communication*, *Asynchronous Batch Jobs*, *Testing*, *Dynamic Form Controller*.

6 Threats to Validity

Internal validity threats, in our study, relate to the authors' bias while conducting the analysis as we have manually labeled the topics. We mitigate the bias in our manual labeling of topics, types of questions, and LCSD phases by consulting the labels among multiple authors and resolving any conflicts via discussion. Four of the authors actively participated in the labelling process. The first author reviewed the final labels and refined the labels by consulting with the second author.

Construct Validity threats relate to the errors that may occur in data collection, like identifying relevant LCSD tags. To mitigate this, we created our initial list of tags, as stated in Section 3, by analyzing the posts in SO related to the leading LCSD platforms. Then we expanded our tag list using state-of-art approach (Bagherzadeh and Khatchadourian 2019; Abdellatif et al. 2020; Ahmed and Bagherzadeh 2018; Rosen and Shihab 2016). Another potential threat is the topic modeling technique, where we choose $K = 45$ as the optimal number of topics for our dataset B . This optimal number of topics has a direct impact on the output of LDA. We experimented with different values of K following related works (Abdellatif et al. 2020; Bagherzadeh and Khatchadourian 2019). We used the coherence score and manual examination to find K 's optimal value that gives us the most relevant and generalized low-code related topics (Alamin et al. 2021; Uddin et al. 2021b; Abdellatif et al. 2020).

External Validity threats relate to the generalizability of our findings. Our study is based on data from developers' discussions on SO. However, there are other forums LCSD developers may use to discuss. We only considered questions and accepted answers in our topic modeling. We also had the option of choosing the best answer. In SO, the accepted answer and best answer may be different. Accepted answer is the one approved by the questioner while the best answer is voted by all the viewers. as discussed in Section 5.1 it is quite difficult to detect if an answer is relevant to the question or not. Thus We chose the accepted answer in this study because we believe that the questioner is the best judge of whether the

answer solves the problem or not. Even without the unaccepted answers, our dataset contains around 38K posts (27K questions + 11K accepted answers). This also conforms with previous works (Uddin et al. 2021b; Abdellatif et al. 2020; Alamin et al. 2021; Yang et al. 2016a). Some novice practitioners post duplicate questions, assign incorrect tags, and provide inadequate descriptions, which receives an overall negative score from the community. To ensure that topics contain relevant discussion of high quality, we only use posts with non-negative scores. Nevertheless, we believe using SO's data provides us with generalizability because SO is a widely used Q&A platform for developers. However, we also believe this study can be complemented by including the best answers to the questions in SO, as we discussed earlier, including discussions from other forums, surveying, and interviewing low-code developers.

7 Related Work

We previously published a paper at the MSR 2021 based on an empirical study of LCSD topics in SO (see (Alamin et al. 2021)). We compared the findings of this paper against our previous paper in Section 1. Other related work can broadly be divided into two categories: SE (Software Engineering) research on/using (1) low code software development (Section 7.1), and (2) topic modeling (Section 7.2).

7.1 Research on Low Code Software Development and Methodologies

LCSD is a new technology, with only a handful of research papers published in this field. Some research has been conducted on the potential applications of this developing technology in various software applications (Fryling 2019) or for automating business process in manufacturing (Waszkowski 2019), healthcare (Ness and Hansen 2019; Woo 2020), Digital transformation (Phalake and Joshi 2021), Industrial engineering education(Adrian et al. 2020), IoT systems using LCSD (Ihirwe et al. 2020a). Di Sipio et al. (2020) present the benefits and future potential of LCSD by sharing their experience of building a custom recommendation system in the LCSD platform. Kourouklidis et al. (2020) discuss the low-code solution to monitor the machine learning model's performance. Sahay et al. (2020) survey LCDP and compare different LCDPs based on their helpful features and functionalities. Khorram et al. (2020) analyse commercial LCSD platforms and present a list of features and testing challenges. Zhuang et al. (2021) created a low-code platform called EasyFL where researchers and educators can easily build systems for privacy-preserving distributed learning method. Ibirwe et al. (2020b) analyse 16 LCSD platforms and identifies what IoT application-related features and services each platform provides. All of these studies compare a single LCSD platform and its support and limitations for various sorts of applications (Alonso et al. 2020), rather than taking a holistic picture of the difficulties that the broader community faces.

There are also some studies where researchers proposed different techniques to improve LCSD platform such as Overeem and Jansen (2021) on LCSD platform's impact analysis, Jacinto et al. (2020) improve testing for LCSD platforms.

Additionally, there are some studies that describe the difficulties faced by LCSD practitioners. The main research methodology and objectives of these studies, however, are significantly different from this study. Luo et al. (2021) analyse the LCSD platform's characteristics including programming languages used, major implementation units, supporting

technologies, applications being developed, domains, etc., along with the benefits, limitations, and challenges by collecting relevant posts from SO and Reddit. In this study, we use tag-based approach to find relevant LCSD-related posts which is much more reliable than text-based searching. Furthermore, the SO related discussion used in this study is significantly larger and our research objective about LCSD platforms challenges are quite different. Lethbridge (2021) discuss experiences with several low-code platforms and provide recommendations focusing on low-code platforms enabling scaling, understandability, documentability, testability, vendor-independence, and the overall user experience for developers as end-users who do some development. Dahlberg (2020) and Alsaadi et al. (2021) Surveyed on factors hindering the widespread adaptation of LCSD by interviewing LCSD developers or conducting a survey. To the best of our knowledge, ours is the first empirical study of LCSD platforms based on developers' discussions from Stack Overflow, and hence our findings complement those of other studies.

7.2 Topic Modeling in Software Engineering

Our motivation to use topic modeling to understand LCSD discussions stems from existing research in software engineering that shows that topics generated from textual contents can be a good approximation of the underlying *themes* (Chen et al. 2016; Sun et al. 2015b, 2016). Topic models are used recently to understand software logging (Li et al. 2018) and previously for diverse other tasks, such as concept and feature location (Cleary et al. 2009; Poshyvanyk et al. 2007), traceability linking (e.g., bug) (Rao and Kak 2011; Asuncion et al. 2010), to understand software and source code history evolution (Hu et al. 2015; Thomas et al. 2011, 2014), to facilitate code search by categorizing software (Tian et al. 2009), to refactor software code base (Bavota et al. 2014), as well as to explain software defect (Chen et al. 2012), and various software maintenance tasks (Sun et al. 2015a, b). The SO posts are subject to several studies on various aspects of software development using topic modeling, such as what developers are discussing in general (Barua et al. 2014) or about a particular aspect, e.g., concurrency (Ahmed and Bagherzadeh 2018), big data (Bagherzadeh and Khatchadourian 2019), chatbot (Abdellatif et al. 2020).

In particular, SO posts have been used in various studies where the researchers analysed topics for that particular domains. For instance, SO posts has been used to study developers challenges in IoT (Uddin et al. 2021b), big data (Bagherzadeh and Khatchadourian 2019), chatbots (Abdellatif et al. 2020) and so on. The distributions and the nature of these posts differs. As SO is arguably the most popular public forum for developers, the analysis of these domains' characteristics may help us identify the SO community better. Therefore, a systematic analysis of these domains is interesting. Following related studies(Uddin et al. 2021b), we use six metrics in this study: (1) Total #posts, (2) Avg views, (3) Avg favorite, (4) Avg score, (5) Percentage of questions without accepted answers, (6) Median hours to get accepted answers per domain. The first four metrics are popularity metrics and the last two are difficulty metrics.

In this study, we do not replicate the findings of the original study in our dataset. Rather we only report the findings from the original study. So following related work (Uddin et al. 2021b), we compared our LCSD-related discussions with other five domains: IoT (Uddin et al. 2021b), big data (Bagherzadeh and Khatchadourian 2019), security (Yang et al. 2016b), mobile apps (Rosen and Shihab 2015), chatbots (Abdellatif et al. 2020) and concurrency (Ahmed and Bagherzadeh 2018).

Table 8 provides an overview of the seven metrics. We can see that it has a greater number of SO posts than chatbot domains but fewer than the other five domains. There are two

Table 8 Comparing the popularity and difficulty metrics of different domains

Type	Metrics	LCSD	IoT	Big data	Chatbot	Security	Mobile	Concurrency
P	# Posts	33,766	53,173	125,671	3,890	94,541	1,604,483	245,541
	Avg View	1330.6	1,320.3	1,560.4	512.4	2,461.1	2,300.0	1,641
	Avg Favorite	1.2	1.5	1.9	1.6	3.8	2.8	0.8
	Avg Score	0.7	0.8	1.4	0.7	2.7	2.1	2.5
D	% W/o Acc. Ans	41%	64%	60.3%	67.7%	48.2%	52%	43.8%
	Med Hrs to Acc.	5.7	2.9	3.3	14.8	0.9	0.7	0.7

other studies on Blockchain (Wan et al. 2019) and deep learning (Han et al. 2020) where the total number of posts are 32,375 and 25,887, respectively. However, these two studies did not report the other metrics, so they are excluded from the Table. Although the LCSD-related discussion may have fewer posts than these other domains, as discussed in RQ3, this number is increasing rapidly.

We can also observe that the LCSD domain shows similarities with IoT, Concurrency domain in terms of Avg. View count. Security and Mobile domain seem most popular in terms of Avg. Favourite count and LCSD rank lower in this metric. LCSD domains most resemble with IoT domain in terms of Avg. View, Avg. Favourite and Avg. Score. In terms of difficulty metrics percentage of posts without accepted answers, LCSD domain ranks lowest, which is good. However, it takes much longer to get accepted answers(0.7 hours for Mobile, 2.9 for IoT). Only the chatbot domain requires more time to get accepted answers (i.e., 5.3 hours in LCSD vs 14.8 hours in chatbot).

We further discuss the LCSD, IoT, and Chatbot domains with the distribution of different types of questions in Table 9. We find that LCSD domain is in the middle of IoT and Chatbot domains in terms of How-type (57%) compared to IoT (47%) and chatbot (62%). This signifies that LCSD domain practitioners ask more about implementation-related questions. In the Why-type question percentage of LCSD-related domains is lowest (14%) compared with IoT (20%) and chatbot (25%). This suggests that practitioners in the LCSD domain enquire about relatively modest troubleshooting issues. For What-type, we notice that the IoT domain dominates with 38% of questions, compared to the LCSD domain's 12%. Practitioners of LCSD are less inquisitive about domain architecture and technologies compared to IoT domain. As a result of these analyses, we can see that LCSD domain practitioners exhibit several traits that distinguish them from practitioners in other domains, which LCSD vendors and educators should take into account.

Table 9 Comparative analysis of question types across different domains

Question type	How	What	Why	Others
LCSD	55.6%	17.9%	14%	12.5%
IoT	47.3%	37.9%	20%	8.3%
Chatbot	61.8%	11.7%	25.4%	1.2%

8 Conclusions

Low Code Software Development (LCSD) is a novel paradigm for developing software applications utilizing visual programming with minimum hand-coding. We present an empirical study that provides insights into the types of discussions low-code developers discuss in Stack Overflow (SO). We find 40 low-code topics in our dataset of 33.7K SO posts (question + accepted answers). We collected these posts based on 64 SO tags belonging to the popular 38 LCSD platforms. We categorize them into five high-level groups, namely Application Customization (30% Questions, 11 Topics), Data Storage (25% Questions, 9 Topics), Platform Adoption (20% Questions, 9 Topics), Platform Maintenance (14% Questions, 6 Topics), and Third-Party Integration (12% Questions, 5 Topics). We find that the Platform Adoption topic category has gained popularity recently. Platform Related Query and Message Queue topics from this category are the most popular. On the other hand, We also find that practitioners find Platform Adoption and Maintenance related topics most challenging. How-type questions are the most common, but our research reveals that practitioners find what-type and why-type questions more difficult. Despite extensive support for testing, deployment, and maintenance, our analysis shows that server configuration, data migration, and system module upgrading-related queries are widespread and complex to LCSD practitioners. Despite significant testing, deployment, and maintenance support, our analysis finds numerous and complex queries regarding server configuration, data migration, and system module updating. Our analysis finds that better tutorial-based documentation can help solve many of these problems. We also find that during the Covid-19 pandemic, LCSD platforms were very popular with developers, especially when it came to dynamic form-based applications. We hope that all of these findings will help various LCSD stakeholders (e.g., LCSD platform vendors, practitioners, SE researchers) to take necessary actions to address the various LCSD challenges. Since the growth indicates that this technology is likely to be widely adopted by various companies for their internal and customer-facing applications, platform providers should address the prevailing developers' challenges. Our future work will focus on (1) getting developers' feedback on our study findings based on surveys and developer interviews, and (2) developing tools and techniques to automatically address the challenges in the LCSD platforms that we observed.

References

- Abdellatif A, Costa D, Badran K, Abdalkareem R, Shihab E (2020) Challenges in chatbot development: a study of stack overflow posts. In: Proceedings of the 17th international conference on mining software repositories, MSR '20. Association for Computing Machinery, New York, pp 174–185
- Adrian B, Hinrichsen S, Nikolenko A (2020) App development via low-code programming as part of modern industrial engineering education. In: International conference on applied human factors and ergonomics. Springer, pp 45–51
- Agrawal A, Fu W, Menzies T (2018) What is wrong with topic modeling? And how to fix it using search-based software engineering. *Inf Softw Technol* 98:74–88
- Ahmed S, Bagherzadeh M (2018) 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, ESEM '18. Association for Computing Machinery, New York
- Akiki PA, Akiki PA, Bandara AK, Yu Y (2020) Eud-mars: end-user development of model-driven adaptive robotics software systems. *Sci Comput Program* 200:102534
- Alamin MAA, Malakar S, Uddin G, Afroz S, Haider TB, Iqbal A (2021) An empirical study of developer discussions on low-code software development challenges. In: 2021 IEEE/ACM 18th international conference on mining software repositories (MSR). IEEE, pp 46–57

- Alonso AN, Abreu J, Nunes D, Vieira A, Santos L, Soares T, Pereira J (2020) Towards a polyglot data access layer for a low-code application development platform. arXiv:[2004.13495](https://arxiv.org/abs/2004.13495)
- Alsaadi HA, Radain DT, Alzahrani MM, Alshammari WF, Alahmadi D, Fakieh B (2021) Factors that affect the utilization of low-code development platforms: survey study. Romanian J Inf Sci Autom Control 31(3):123–140
- Amazon Honeycode Platform Overview (2022) Available: <https://www.honeycode.aws/>. (Online; accessed 5-January-2022)
- App Engine (2021) A fully managed, serverless platform for developing and hosting web applications at scale. Available: <https://cloud.google.com/appengine/docs>. (Online; accessed 13-December-2021)
- AppSheet (2021) Low-code application development. Available: <https://www.appsheet.com>. (Online; accessed 13-December-2021)
- Appian Platform Overview (2022) Available: <https://www.appian.com/>. (Online; accessed 5-January-2022)
- Arun R, Suresh V, Madhavan CV, Murthy MN (2010) On finding the natural number of topics with latent dirichlet allocation: Some observations. In: Pacific-asia conference on knowledge discovery and data mining. Springer, pp 391–402
- Asaduzzaman M, Mashiyat AS, Roy CK, Schneider KA (2013) Answering questions about unanswered questions of stack overflow. In: 2013 10th Working conference on mining software repositories (MSR). IEEE, pp 97–100
- Asuncion HU, Asuncion AU, Taylor RN (2010) Software traceability with topic modeling. In: 2010 ACM/IEEE 32nd international conference on software engineering, vol 1. IEEE, pp 95–104
- AWS Amplify Studio Overview (2022) Available: <https://aws.amazon.com/amplify/studio/>. (Online; accessed 5-January-2022)
- Bagherzadeh M, Khatchadourian R (2019) Going big: a large-scale study on what big data developers ask. In: Proceedings of the 2019 27th ACM joint meeting on European softwa engineering conference and symposium on the foundations of software engineering, ESEC/FSE 2019. ACM, New York, pp 432–442
- Bajaj K, Pattabiraman K, Mesbah A (2014) Mining questions asked by web developers. In: Proceedings of the 11th working conference on mining software repositories, pp 112–121
- Bandeira A, Medeiros CA, Paixao M, Maia PH (2019) We need to talk about microservices: an analysis from the discussions on stackoverflow. In: 2019 IEEE/ACM 16th international conference on mining software repositories (MSR). IEEE, pp 255–259
- Barua A, Thomas SW, Hassan AE (2014) What are developers talking about? An analysis of topics and trends in stack overflow. Empir Softw Eng 19(3):619–654
- Basciani F, Iovino L, Pierantonio A et al (2014) Mdforge: an extensible web-based modeling platform. In: 2nd International workshop on model-driven engineering on and for the cloud, cloudMDE 2014, co-located with the 17th international conference on model driven engineering languages and systems, moDELS 2014, vol 1242. CEUR-WS, pp 66–75
- Basil VR, Turner AJ (1975) Iterative enhancement: a practical technique for software development. IEEE Trans Softw Eng (4):390–396
- Bavota G, Oliveto R, Gethers M, Poshyvanyk D, Lucia AD (2014) Methodbook: recommending move method refactorings via relational topic models. IEEE Trans Softw Eng 40(7):671–694
- Bayer J, Muthig D (2006) A view-based approach for improving software documentation practices. In: 13th Annual IEEE international symposium and workshop on engineering of computer-based systems (ECBS'06). IEEE, p 10pp
- Beck K, Beedle M, Van Bennekum A, Cockburn A, Cunningham W, Fowler M, Grenning J, Highsmith J, Hunt A, Jeffries R et al (2001) Manifesto for agile software development
- Beynon-Davies P, Carne C, Mackay H, Tudhope D (1999) Rapid application development (rad): an empirical review. Eur J Inf Syst 8(3):211–223
- Bhat JM, Gupta M, Murthy SN (2006) Overcoming requirements engineering challenges: lessons from offshore outsourcing. IEEE Softw 23(5):38–44
- Blei DM, Ng AY, Jordan MI (2003) Latent dirichlet allocation. J Mach Learn Res 3(4-5):993–1022
- Botterweck G (2006) A model-driven approach to the engineering of multiple user interfaces. In: International conference on model driven engineering languages and systems. Springer, pp 106–115
- Brambilla M, Cabot J, Wimmer M (2017) Model-driven software engineering in practice. Synth Lect Comput Sci 3(1):1–207
- Brambilla M, Umohoza E, Acerbis R (2017) Model-driven development of user interfaces for iot systems via domain-specific components and patterns. J Internet Serv Appl 8(1):1–21
- Burnett MM, McIntyre DW (1995) Visual programming. COmputer-Los Alamitos- 28:14–14
- Chakraborty P, Shahriyar R, Iqbal A, Uddin G (2021) How do developers discuss and support new programming languages in technical q&a site? An empirical study of go, swift, and rust in stack overflow. Inf Softw Technol (IST) 19

- Chen T-H, Thomas SW, Nagappan M, Hassan AE (2012) Explaining software defects using topic models. In: 9th Working conference on mining software repositories, pp 189–198
- Chen T-HP, Thomas SW, Hassan AE (2016) A survey on the use of topic models when mining software repositories. *Empir Softw Eng* 21(5):1843–1919
- Cleary B, Exton C, Buckley J, English M (2009) An empirical analysis of information retrieval based concept location techniques in software comprehension. *Empir Softw Eng* 14:93–130
- Costabile MF, Fogli D, Mussio P, Piccinno A (2007) Visual interactive systems for end-user development:, a model-based design methodology. *IEEE Trans Syst Man Cybern: Syst Part A: Syst Hum* 37(6):1029–1046
- Dahlberg D (2020) Developer experience of a low-code platform: an exploratory study
- De Lucia A, Di Penta M, Oliveto R, Panichella A, Panichella S (2014) Labeling source code with information retrieval methods: an empirical study. *Empir Softw Eng* 19(5):1383–1420
- Di Sipio C, Di Ruscio D, Nguyen PT (2020) Democratizing the development of recommender systems by means of low-code platforms. In: Proceedings of the 23rd ACM/IEEE international conference on model driven engineering languages and systems: companion proceedings, pp 1–9
- Enterprise Low-Code Application Platforms (LCAP) Reviews and Ratings (2022) Available: <https://www.gartner.com/reviews/market/enterprise-low-code-application-platform>. (Online; accessed 5-January-2022)
- Exchange S (2020) Stack exchange data dump. Available: <https://archive.org/details/stackexchange>. (Online; accessed 5-January-2022)
- Fincher S, Tenenberg J (2005) Making sense of card sorting data. *Expert Syst* 22(3):89–93
- Fischer G, Giaccardi E, Ye Y, Sutcliffe AG, Mehandjiev N (2004) Meta-design: a manifesto for end-user development. *Commun ACM* 47(9):33–37
- Fors N (2016) The design and implementation of Bloqqi-a feature-based diagram programming language. PhD thesis, Lund University
- Fryling M (2019) Low code app development. *J Comput Sci Coll* 34(6):119
- Google App Maker Platform Overview (2020) Available: <https://developers.google.com/appmaker>. (Online; accessed 5-January-2022)
- Google App Maker will be shut down on January 19 2021 (2021) <https://workspaceupdates.googleblog.com/2020/01/app-maker-update.html>. (Online; accessed 5-January-2022)
- Halbert DC (1984) Programming by example. PhD thesis, University of California, Berkeley
- Han J, Shihab E, Wan Z, Deng S, Xia X (2020) What do programmers discuss about deep learning frameworks. *Empir Softw Eng* 25(4):2694–2747
- How many Low-Code/No-Code platforms are out there? (2022) Available: <https://www.spreadsheetweb.com/how-many-low-code-no-code-platforms-are-out-there/>. (Online; accessed 5-August-2022)
- Hu J, Sun X, Lo D, Li B (2015) Modeling the evolution of development topics using dynamic topic models. In: IEEE 22nd international conference on software analysis, evolution, and reengineering, pp 3–12
- IBM Lotus Software (2022) Available: <https://help.hcltechsw.com/>. (Online; accessed 5-January-2022)
- Ihirwe F, Di Ruscio D, Mazzini S, Pierini P, Pierantonio A (2020a) Low-code engineering for internet of things: a state of research. In: Proceedings of the 23rd ACM/IEEE international conference on model driven engineering languages and systems: companion proceedings, pp 1–8
- Ihirwe F, Di Ruscio D, Mazzini S, Pierini P, Pierantonio A (2020b) Low-code engineering for internet of things: a state of research. In: Proceedings of the 23rd ACM/IEEE international conference on model driven engineering languages and systems: companion proceedings, MODELS '20. Association for Computing Machinery, New York
- Jacinto A, Lourenço M, Ferreira C (2020) Test mocks for low-code applications built with outsystems. In: Proceedings of the 23rd ACM/IEEE international conference on model driven engineering languages and systems: companion proceedings, pp 1–5
- Kendall MG (1938) A new measure of rank correlation. *Biometrika* 30(1):81–93
- Khan JY, Khondaker MTI, Uddin G, Iqbal A (2021a) Automatic detection of five api documentation smells: Practitioners' perspectives. In: 2021 IEEE International conference on software analysis, evolution and reengineering (SANER). IEEE, pp 318–329
- Khan JY, Khondaker MTI, Uddin G, Iqbal A (2021b) Automatic detection of five api documentation smells: Practitioners' perspectives. In: IEEE International conference on software analysis, evolution and reengineering (SANER), p 12
- Khorram F, Mottu J-M, Sunyé G (2020) Challenges & opportunities in low-code testing. In: Proceedings of the 23rd ACM/IEEE international conference on model driven engineering languages and systems: companion proceedings, MODELS '20. Association for Computing Machinery, New York

- Kourouklidis P, Kolovos D, Matragkas N, Noppen J (2020) Towards a low-code solution for monitoring machine learning model performance. In: Proceedings of the 23rd ACM/IEEE international conference on model driven engineering languages and systems: companion proceedings, pp 1–8
- Kruskal WH (1957) Historical notes on the wilcoxon unpaired two-sample test. *J Am Stat Assoc* 52(279):356–360
- Lethbridge TC (2021) Low-code is often high-code, so we must design low-code platforms to enable proper software engineering. In: International symposium on leveraging applications of formal methods. Springer, pp 202–212
- Li H, Chen T-HP, Shang W, Hassan AE (2018) Studying software logging using topic models. *Empir Softw Eng* 23:2655–2694
- Lin G, Wen S, Han Q-L, Zhang J, Xiang Y (2020) Software vulnerability detection using deep neural networks: a survey. *Proc IEEE* 108(10):1825–1848
- Linares-Vásquez M, Dit B, Poshyvanyk D (2013) An exploratory analysis of mobile development issues using stack overflow. In: 2013 10th Working conference on mining software repositories (MSR). IEEE, pp 93–96
- Loper E, Bird S (2002) NLTK: the natural language toolkit. arXiv:cs/0205028
- Low-Code Development Platform (2022) Available: https://en.wikipedia.org/wiki/Low-code_development_platform. (Online; accessed 5-January-2022)
- Luo Y, Liang P, Wang C, Shahin M, Zhan J (2021) Characteristics and challenges of low-code development: the practitioners' perspective. In: Proceedings of the 15th ACM/IEEE international symposium on empirical software engineering and measurement (ESEM), pp 1–11
- McCallum AK (2002) Mallet: a machine learning for language toolkit. <http://mallet.cs.umass.edu>
- McHugh ML (2012) Interrater reliability: the kappa statistic. *Biochem Medica* 22(3):276–282
- Mendix Platform Overview (2022) Available: <https://www.mendix.com/>. (Online; accessed 5-January-2022)
- Mernik M, Heering J, Sloane AM (2005) When and how to develop domain-specific languages. *ACM Comput Surv (CSUR)* 37(4):316–344
- Microsoft Power Apps Platform Overview (2022) Available: <https://powerapps.microsoft.com/en-us/>. (Online; accessed 5-January-2022)
- Microsoft Power FX (2021) Available: https://docs.microsoft.com/en-us/power-platform/power-fx_overview. (Online; accessed 13-December-2021)
- Myers BA, Ko AJ, Burnett MM (2006) Invited research overview: end-user programming. In: CHI'06 Extended abstracts on human factors in computing systems, pp 75–80
- Ness C, Hansen ME (2019) Potential of low-code in the healthcare sector: an exploratory study of the potential of low-code development in the healthcare sector in Norway. Master's thesis
- OneBlink Platform Overview (2022) Available: <https://www.oneblink.io/>. (Online; accessed 5-January-2022)
- Oracle Apex Platform (2022) Available: <https://apex.oracle.com/>. (Online; accessed 5-January-2022)
- Overeem M, Jansen S (2021) Proposing a framework for impact analysis for low-code development platforms. In: 2021 ACM/IEEE International conference on model driven engineering languages and systems companion (MODELS-c). IEEE, pp 88–97
- Overflow S (2020) Stack Overflow Questions. <https://stackoverflow.com/questions/>. Last accessed on 14 November 2020
- Pane J, Myers B (2006) More natural programming languages and environments. Springer, pp 31–50
- Paternò F (2013) End user development: survey of an emerging field for empowering people. *International Scholarly Research Notices*
- Phalake VS, Joshi SD (2021) Low code development platform for digital transformation. In: Information and communication technology for competitive strategies (ICTCS 2020). Springer, pp 689–697
- Pleuss A, Wollny S, Botterweck G (2013) Model-driven development and evolution of customized user interfaces. In: Proceedings of the 5th ACM SIGCHI symposium on engineering interactive computing systems, pp 13–22
- Ponzanelli L, Mocci A, Bacchelli A, Lanza M, Fullerton D (2014) Improving low quality stack overflow post detection. In: 2014 IEEE International conference on software maintenance and evolution. IEEE, pp 541–544
- Poshyvanyk D, Guéhéneuc Y-G, Marcus A, Antoniol G, Rajlich VT (2007) Feature location using probabilistic ranking of methods based on execution scenarios and information retrieval. *IEEE Trans Softw Eng* 33(6):420–432
- Programming Gains Speed as Developers Turn to Low-Code During the Pandemic (2022) Available: <https://www.designnews.com/automation/programming-gains-speed-developers-turn-low-code-during-pandemic>. (Online; accessed 5-August-2022)
- Quickbase Platform Overview (2022) Available: <https://www.quickbase.com/product/product-overview>. (Online; accessed 5-January-2022)

- Ramasubramanian C, Ramya R (2013) Effective pre-processing activities in text mining using improved porter's stemming algorithm. *Int J Adv Res Comput Commun Eng* 2(12):4536–4538
- Rao S, Kak AC (2011) Retrieval from software libraries for bug localization: a comparative study of generic and composite text models. In: 8th Working conference on mining software repositories, pp 43–52
- Rehurek R, Sojka P (2010) Software framework for topic modelling with large corpora. In: Proceedings of the LREC 2010 workshop on new challenges for NLP frameworks. Citeseer
- Ren X, Xing Z, Xia X, Li G, Sun J (2019) Discovering, explaining and summarizing controversial discussions in community q&a sites. In: 2019 34th IEEE/ACM international conference on automated software engineering (ASE). IEEE, pp 151–162
- Resnick M, Maloney J, Monroy-Hernández A, Rusk N, Eastmond E, Brennan K, Millner A, Rosenbaum E, Silver J, Silverman B et al (2009) Scratch: programming for all. *Commun ACM* 52(11):60–67
- Robillard MP, Bodden E, Kawrykow D, Mezini M, Ratchford T (2012) Automated API property inference techniques. *IEEE Trans Softw Eng* 28
- Röder M, Both A, Hinneburg A (2015) Exploring the space of topic coherence measures. In: Proceedings of the eighth ACM international conference on Web search and data mining, pp 399–408
- Rosen C, Shihab E (2015) What are mobile developers asking about? A large scale study using stack overflow. *Empir Softw Eng* 33
- Rosen C, Shihab E (2016) What are mobile developers asking about? A large scale study using stack overflow. *Empir Softw Eng* 21(3):1192–1223
- Rymer JR, Koplowitz R, Leaders SA (2019) The forrester wave(tm) low-code development platforms for ad&d professionals, vol q1, 2019
- Sahay A, Indamutsa A, Di Ruscio D, Pierantonio A (2020) Supporting the understanding and comparison of low-code development platforms. In: 2020 46th Euromicro conference on software engineering and advanced applications (SEAA). IEEE, pp 171–178
- Salesforce Platform Overview (2022) Available: <https://www.salesforce.com/in/?ir=1>. (Online; accessed 5-January-2022)
- Sinha G, Shahi R, Shankar M (2010) Human computer interaction. In: 2010 3rd International conference on emerging trends in engineering and technology. IEEE, pp 1–4
- Sun X, Li B, Leung H, Li B, Li Y (2015a) Msr4sm: using topic models to effectively mining software repositories for software maintenance tasks. *Inf Softw Technol* 66:671–694
- Sun X, Li B, Li Y, Chen Y (2015b) What information in software historical repositories do we need to support software maintenance tasks? An approach based on topic model. *Comput Inf Sci* 22–37
- Sun X, Liu X, Li B, Duan Y, Yang H, Hu J (2016) Exploring topic models in software engineering data analysis: a survey. In: 17th IEEE/ACIS international conference on software engineering, artificial intelligence, networking and parallel/distributed computing, pp 357–362
- The Best Low-Code Development Platforms (2022) Available: <https://www.pcmag.com/picks/the-best-low-code-development-platforms>. (Online; accessed 5-January-2022)
- Thomas SW, Adams B, Hassan AE, Blostein D (2011) Modeling the evolution of topics in source code histories. In: 8th Working conference on mining software repositories, pp 173–182
- Thomas SW, Adams B, Hassan AE, Blostein D (2014) Studying software evolution using topic models. *Sci Comput Program* 80(B):457–479
- Tian K, Revelle M, Poshyvanyk D (2009) Using latent dirichlet allocation for automatic categorization of software. In: 6th International working conference on mining software repositories, pp 163–166
- Torres C (2018) Demand for programmers hits full boil as us job market simmers. Bloomberg Com
- Treude C, Barzilay O, Storey M-A (2011) How do programmers ask and answer questions on the web?(tier track). In: Proceedings of the 33rd international conference on software engineering, pp 804–807
- Uddin G, Khomh F (2017a) Automatic summarization of api reviews. In: 2017 32nd IEEE/ACM international conference on automated software engineering (ASE). IEEE, pp 159–170
- Uddin G, Khomh F (2017b) Automatic summarization of API reviews. In: Proceedings of the 32nd IEEE/ACMinternational conference on automated software engineering, pp 12
- Uddin G, Khomh F (2017c) Opiner: a search and summarization engine for API reviews. In: Proceedings of the 32nd IEEE/ACM international conference on automated software engineering, pp 6
- Uddin G, Khomh F (2019) Automatic opinion mining from API reviews from stack overflow. *IEEE Trans Softw Eng* 35
- Uddin G, Robillard MP (2015a) How api documentation fails. *IEEE Softw* 32(4):68–75
- Uddin G, Robillard MP (2015b) How api documentation fails. *IEEE Softw* 32(4):76–83
- Uddin G, Baysal O, Guerroj L, Khomh F (2019) Understanding how and why developers seek and analyze api related opinions. *IEEE Trans Softw Eng* 40
- Uddin G, Khomh F, Roy CK (2020a) Automatic api usage scenario documentation from technical q&a sites. *ACM Trans Softw Eng Methodol* 43

- Uddin G, Khomh F, Roy CK (2020b) Automatic mining of api usage scenarios from stack overflow. *Inf Softw Technol (IST)* 16
- Uddin G, Khomh F, Roy CK (2021a) Automatic api usage scenario documentation from technical q&a sites. *ACM Trans Softw Eng Methodol (TOSEM)* 30(3):1–45
- Uddin G, Sabir F, Guéhéneuc Y-G, Alam O, Khomh F (2021b) An empirical study of iot topics in iot developer discussions on stack overflow. *Empir Softw Eng* 26:11
- Ur B, McManus E, Pak Yong Ho M, Littman ML (2014) Practical trigger-action programming in the smart home. In: Proceedings of the SIGCHI conference on human factors in computing systems, pp 803–812
- Van Deursen A, Klint P, Visser J (2000) Domain-specific languages: an annotated bibliography. *ACM Sigplan Not* 35(6):26–36
- Vincent P, Lijima K, Driver M, Wong J, Natis Y (2019) Magic quadrant for enterprise low-code application platforms. Retrieved December 18, 2019
- Vinyl Platform Overview (2022) Available: <https://zudy.com/>. (Online; accessed 5-January-2022)
- Wan Z, Xia X, Hassan AE (2019) What is discussed about blockchain? A case study on the use of balanced lda and the reference architecture of a domain to capture online discussions about blockchain platforms across the stack exchange communities. *IEEE Transactions on Software Engineering*
- Waszkowski R (2019) Low-code platform for automating business processes in manufacturing. *IFAC-PapersOnLine* 52:376–381
- Wolber D (2011) App inventor and real-world motivation. In: Proceedings of the 42nd ACM technical symposium on computer science education, pp 601–606
- Wong J, Driver M, Vincent P (2019) Low-code development technologies evaluation guide
- Woo M (2020) The rise of no/low code software development—no experience needed? *Engineering (Beijing China)* 6(9):960
- Yang D, Hussain A, Lopes CV (2016a) From query to usable code: an analysis of stack overflow code snippets. In: 2016 IEEE/ACM 13th working conference on mining software repositories (MSR). IEEE, pp 391–401
- Yang X-L, Lo D, Xia X, Wan Z-Y, Sun J-L (2016b) What security questions do developers ask? A large-scale study of stack overflow posts. *J Comput Sci Technol* 31(5):910–924
- Zhu L, Bass L, Champlin-Scharff G (2016) Devops and its practices. *IEEE Softw* 33(3):32–34
- Zhuang W, Gan X, Wen Y, Zhang S (2021) Easyfl: a low-code federated learning platform for dummies. arXiv:2105.07603
- Zoho Creator Platform Overview (2022) Available: <https://www.zoho.com/creator/>. (Online; accessed 5-January-2022)



Md Abdullah Al Alamin received his B.Sc. degree from Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh in 2017 in Computer Science and Engineering. Later he worked as a software engineer at Reve Systems Ltd and Samsung R&D Institute Bangladesh. Currently he is doing his master's in software engineering under the supervision of Dr Gias Uddin at DISA (Data Intensive Software Analytics) lab at the University of Calgary, Canada. His research interests include empirical study and applying state of the art machine learning solutions in software engineering domain. His current research focuses on a low-code approach to democratize software development and machine learning to domain experts.



Gias Uddin is a professor at the University of Calgary from July 2020, where he is the founding director of the Data Intensive Software Analytics (DISA) lab. His research lies at the intersection of software engineering and machine learning, to understand and to facilitate how the two domains can benefit from each other. At the DISA lab, he leads his students to work on software security, responsible AI engineering of machine learning software, and software analytics focusing on software quality. Dr. Uddin received a PhD from McGill University and a master's from Queen's University (Canada). Dr. Uddin spent 10+ years full-time working in four software and government companies before, during, and after his PhD. Prior to joining University of Calgary, Dr. Uddin was a Senior Data Scientist at the Bank of Canada (the central Bank of Canada) and a software engineer at IBM Watson Analytics. In various increasingly senior roles in the Industry, Dr. Uddin has designed and developed software using innovative machine learning techniques. The software is deployed successfully in production and is being used by thousands/millions of users.

Dr. Uddin's PhD research was funded by several fellowships, including a three-year PhD fellowship from IBM. He has published more than 35 papers in peer-reviewed topmost conferences and journals in software engineering, including International Conference on Software Engineering (ICSE), IEEE Transactions on Software Engineering (TSE), ACM Transactions on Software Engineering and Methodology (TOSEM). His recent paper at the 32nd IEEE/ACM Automated Software Engineering Conference (ASE) was nominated for a best paper award.



Sanjay Malakar received his Bachelor of Science degree in Computer Science and Engineering from Bangladesh University of Engineering and Technology in 2021. He completed his undergraduate thesis under the supervision of professor Dr. Rifat Shahriyar on addressing atomicity bugs in software. Besides that, he worked with Dr. Gias Uddin and Dr. Anindya Iqbal on an empirical study of low-code software development challenges. His research focuses on Software Engineering, Computer/Software Security, and Programming Languages. Currently, he is working as a software engineer at OpenRefactory, Inc., a Silicon Valley-based deep-tech software company that focuses on automatically fixing security, reliability, and compliance bugs.



Sadia Afroz received her Bachelor of Science degree in Computer Science and Engineering from Bangladesh University of Engineering and Technology in 2021. Currently, she is working as a software engineer at Therap (BD) Ltd. Her research interest consists of Empirical Software Engineering and Human Computer Interaction.



Tameem Haider received his Bachelor of Science degree in Computer Science and Engineering from Bangladesh University of Engineering and Technology in 2021. Currently, He is working as a software engineer at Dynamic Solution Innovators Ltd in Dhaka. His research interests include Empirical Software Engineering, Functional Programming Languages, and Human Computer Interaction.



Anindya Iqbal received B.Sc. Eng. (hons.) and M.Sc. Eng degrees in Computer Science and Engineering from Bangladesh University of Engineering and Technology (BUET), Bangladesh, in 2005 and 2009, respectively. He has received PhD degree from Monash University, Australia in 2013. Currently, he is a professor in the department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET). His major research interests are in the fields of Software Engineering, Empirical Software Engineering, Applied Machine Learning, Security & Privacy, Participatory Sensing System, and Wireless Sensor Networks. He has published more than 35 refereed publications in reputed journals and conferences including EMSE, IST, Ad Hoc Networks, JNCA, ASE, ESEM, MSR, SANER, etc. and serves as TPC member in some conferences including the flagship one FSE and a reviewer of IST. He received research grant from Samsung Research Bangladesh. He also serves as a member of the consultancy team of the Department of CSE, BUET for some of the flagship ICT projects of the Government of Bangladesh.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.