Please quote as: Elshan, E., Bruhin, O., Schmidt, N., Simeon, D. & Kedziora, D. (2024). Unveiling Challenges and Opportunities in Low Code Development Platforms: A StackOverflow Analysis. Proceedings of the 57th Hawaii International Conference on System Sciences (HICSS), O'ahu, Hawaii.

Unveiling Challenges and Opportunities in Low Code Development Platforms: A StackOverflow Analysis

Elshan Edona, Siemon Dominik, Bruhin Olivia, Kedziora Damian, Schmidt Niklas

Unveiling Challenges and Opportunities in Low Code Development Platforms: A StackOverflow Analysis

Edona Elshan
VU Amsterdam, Netherlands
e.elshan@vu.nl

Olivia Bruhin
University of St.Gallen, Switzerland
Olivia.bruhin@unisg.ch

Niklas Schmidt
University of St.Gallen, Switzerland
niklas.schmidt@student.unisg.ch

Dominik Siemon
LUT University, Finland
dominik.siemon@lut.fi

Damian Kedziora
LUT University, Finland
Kozminski University, Poland
damian.kedziora@lut.fi

Abstract

As the rapidly expanding digital transformations at multiple organizations require development of growing number of software solutions, low code development platforms (LCDPs) started to be widely used by pretrained business users, in such use-cases as process automation and rapid application development. Our study explores the challenges of LCDPs use for developers, by investigating 30 000 of their posts at one of the most prominent fora StackOverflow. It is conducted with text-mining approaches, primarily Latent Dirichlet Allocation (LDA), aiming to identify challenges for users of LCDPs. As they were from the areas of visualization, third-party integration, database and table management, datatype conversion, programming languages, and file handling, we further discussed them to propose possible enhancements for users of LCDPs.

Keywords: Low Code Development Platforms; Latent Dirichlet Allocation; StackOverflow

1. Introduction

As the impact of digital transformation (DT) on organizations is rapidly evolving (Wessel et al., 2021), the need for the quick response and transilient adoption to changing market requirements is growing (Sanchis et al., 2019). However, due to a massive shortage of skilled software developers at the labor market (Danhieux, 2022), many organizations face the challenge of the demand for information systems (IS) that is way higher than what can be provided by their IT department (Waszkowski, 2019).What is more, the predictions that the demand for skilled IR professionals will continue to grow faster than the supply provided by the market (Torres, 2018) ,organizations are forced to consider faster and cheaper ways to adapt to their growing software development needs (Fryling, 2019). On the other hand, the burgeoning realm of technology offers a multitude of avenues for individuals and organizations to enhance their productivity and redefine their creative boundaries. From this perspective, low code development platforms (LCDPs) became a revolutionizing trend on the software development landscape (Kedziora, 2022), aiming to address this challenge by democratizing software development and thus accelerating the development and deployment process (Alamin et al., 2023). LCDPs are usually cloud-native services (often categorized as platform-as-a-service) that enable the development of applications with use of pre-automated and AI-driven design tools. LCDPs are therefore part of a larger trend of technology democratization (Brinker, 2018), referring to any assignment that traditionally required coding but can now be accomplished by a pre-trained business user.

Until now, there are already more than 400 LCDPs (Ugur, 2021) for a wide variety of use cases,

URI: https://hdl.handle.net/10125/107258
978-0-9981331-7-1

including process automation and rapid application development. However, despite its huge potential and growing adoption, LCDPs are not without challenges and risk areas (Elshan et al., 2023). As citizen developers and professional software engineers explore the potentials of LCDPs, they inevitably confront a myriad of issues. Recognizing and understanding these problems is a foundational step. Only when we have a clear picture of these challenges can we move towards crafting effective solutions.Until now, it remains unclear if the aforementioned problem can be resolved with the help of LCDPs or not. Prior studies qualitatively investigated the challenges and obstacles of LCDPs (e.g., Elshan et al., 2023; Prinz et al., 2022). In this work, we focus on problems faced by developers and therefore look at online developer communities to extract those. One of the largest Question & Answer (Q&A) sites is StackOverflow, with around 23 million questions and 12 million registered users (StackOverflow, 2022). So far, posts on StackOverflow have been used to conduct research on blockchain, microservices, or recently low code development (Alamin et al., 2023; Luo et al., 2021).

While the mentioned studies provide a comprehensive overview of LCDPs topics and their prevalence within development phases, our research aims to delve deeper into the underlying challenges developers face, especially in the realms of customization and platform integration. We aim to identify specific pain points within the customization and platform adoption phases, hoping to provide actionable insights for LCDP designers and developers. Our focus is not just to catalog the issues but to understand their root causes, something not extensively covered in the previous research. This would aid in evolving LCDPs to be more intuitive and adaptable, catering to the growing demands of modern software development.

Therefore, we pose the following research question (RQ):

RQ: What are the most prominent problems that developers face when using low-code development platforms?

To answer this research question, we investigate developers' posts on LCDPs on StackOverflow. Therefore, we examined more than 30'000 posts using text-mining approaches. In particular, we adopt Latent Dirichlet Allocation (LDA) to uncover important topics addressed by the developers. We then take the identified issues and propose possible enhancements for users of LCDPs and how the boundaries of the platform can be overcome. Our findings contribute to a better understanding of the actual problems developers face, have practical implications for the use of LCDPs, and help to close the skill gap faced in developer applications.

2. Theoretical Background

LCDPs emerged as a transformative set of tools designed for a diverse range of users, from seasoned programmers to novices with no coding experience (Adrian et al., 2020; Bock & Frank, 2021; Kletti, 2021). Not only do they democratize software development, but also enable the production of high-quality software in a compressed timeframe (Sanchis et al. 2019) Typically, designed as cloud-based, Platform-as-a-Service (PaaS) products- LCDPs create a fertile ground for application development, leveraging preautomated, and AI-driven design tools and visual aids.

Equipped with reusable components and configuration settings, LCDPs substantially reduce time required for manual programming (Khorram et al., 2020). The application logic, user interface, or integrations to various data services are created with the help of user-friendly visual tools and can be supplemented by manual code components if needed (Di Sipio et al., 2020). Users who start to gain their programming and IT skills, yet already have business domain expertise, or so-called "citizen developers," are, therefore, one of the main target groups of such platforms (Tisi et al., 2019).

The terms "low code" and "no code" are often used interchangeably. However, for the purposes of this paper, we distinguish between the two based on the possibility of developing custom code. If no self-written code is possible, we use the term "no code" development (Daniel et al., 2020) and when self-written code is still required, albeit in a simplified or limited form, we refer to it as "low code" development. Regardless, for the scope of our study, we employ "low code" as an overarching term encompassing both low code and no code tools, given that most low code solutions offer the flexibility to access and edit code directly.

In general, the idea of low code is not new, as the approaches in which people without computer science knowledge are enabled to build systems independently have been present for a long time. However, external influences such as the rise of digital platforms have changed the IS development landscape. In consequence, platform-centricity is central to the permeation of low code in work environments. To this end, LCDP vendors (such as OutSystems or Mendix) usually provide various tools to support the application development process from initial ideation and modeling, to implementation and maintenance (Almonte et al., 2020). Furthermore, there is an

emerging generation of tech-savvy, digitally native workforce who already have some of the necessary qualifications. Factors such as increased affinity for technology, consumerization, and advancing digitalization are opening up a whole new target group for the low code movement (Woo, 2020).

Visual tools are usually operated according to the drag-and-drop principle, which can simplify and accelerate the software development process and reduce development costs (Rymer et al., 2019). Thus, LCDPs enable rapid and agile development of new IT artifacts and require low technical understanding, which is often prevalent in business development (Pantelimon et al., 2019). In this sense, this leads to faster development, easier understanding, and a basis for better exchange of feedback and ideas (Waszkowski, 2019). Based on visual, model-driven development techniques and visual application designs, LCPDs make it easier to understand the development of an IT project compared to manual programming techniques (Frank et al., 2021). Low code integrates a combination of approaches and IT trends. Rapid application development (RAD), fourth-generation programming languages (4GL), computer-aided software engineering (CASE) tools, and model-driven engineering (MDE) principles are often mentioned in this context. The low code approach takes these concepts and embeds them in full application lifecycle support (Baumgarten et al., 2020). However, admittedly, the increase in attention has not been matched by comparable breakthroughs in the conceptualization of LCDPs (Bock & Frank, 2021). Thus, LCDPs are of particular interest as a subject of research.

3. Methodology

Our research process consists of five steps: (1) Identify keywords and posts, (2) filter candidate posts, (3) extract data, (4) code data and (5) analyze data. First, we will discuss our data collection process to find StackOverfow posts that are related to LCDPs. In a next step, we discuss our pre-processing and topic modeling steps.

3.1. Data Collection

For our analysis, in June 2022, we collected posts from StackOverflow The contents of the "Post.xml" file were used, which contained information about each post such as the unique ID, type (Question or Answer), title, body, associated tags, creation date, view-count, and so on. Then the general approach was to fetch all LCDP's related posts or questions from this dataset. This was conducted by filtering the tags of the posts according to a list of predefined tags. For this step, we created a list for posts that contain tags such as "low code". To find relevant tags, we first compiled a list of LCDPs by assessing platforms of market leaders by Gartner (Vincent et al., 2019), Forrester (Rymer et al., 2019), related research work (Sahay et al., 2020), and other online resources. Our compiled list contained 34 platforms such as Mendix, Microsoft Power Platform and Appian. This list was evaluated and discussed iteratively by two authors before being completed. In a last step, we extracted the LCDPs related posts from the dataset based on the tag list. For each post, it includes body, title and metadata (e.g., CreationDate, ViewCount, Tags, CommentCount etc.).

3.2. Topic Modelling with LDA

One of the most powerful text mining techniques for discovering latent data and discovering relationships in text corpora is topic modeling (Jelodar et al., 2019). The Latent Dirichlet Allocation (LDA) method is the most widely used topic modeling method, and it is used in many scientific disciplines and other practices to identify the most relevant and frequently mentioned topics in specific texts (Asmussen & Møller, 2019; Jelodar et al., 2019). The idea behind LDA is that" the documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words" (Blei et al., 2003, p. 996). To model the topics, we took three actions. First, we pre-processed the posts, then we calculated the optimal number of topics. Lastly, we generated the topics.

3.2.1. Pre-Processing. For the cleaning of the data, we followed the guidelines suggested by Albon (2018) and started by removing noise (i.e., punctuation and white-spaces). We eliminated the platform's name (i.e., PowerBi, Mendix etc.) from the dataset since previous work (Al Alamin et al., 2021) has shown that the resulting topics sometimes are grouped around platforms rather than the technical hurdles addressed. Furthermore, we tokenized the text. In this realm, tokenization is a fundamental step in NLP as it splits the text into words or sentences (Sun et al., 2017). This allows to handle individual words from a text. The tokenization can be done for instance, by using the word tokenize library from nltk, Stanford Tokenizer or OpenNLP Tokenizer (Bird, 2006, Sun et al., 2017). On top of the suggestions from Albon (2018), Sun et al. (2017) point out that nltk provides a stemming library. Stemmers remove morphological affixes from tokens resulting only in the word stem (Bird, 2006). This is

especially helpful when dealing with probabilities based on word occurrences, which might be the case for development or developing; and thus teaches the algorithm to treat the two words as one. For this reason we chose to work with nltk. After stemming our posts, we iterated through the posts and wrote the "*Body*" content of the top posts per topic into a .txt file, which was later used for the analysis. To provide a concrete example of our pre-processing steps, we will dissect a representative post on StackOverflow: "*I've recently started working with a LCDP and am running into some issues with data binding. I have a form component on a page, and I'm trying to bind it to a data object that I've defined in the platform. However, every time I try to submit the form, the data doesn't seem to be updating*."

- Noise Removal: Our first step cleans out any superfluous information or symbols. This would refine our example post to: "I've recently started working with LCDP and am running into some issues with data binding."
- Tokenization: This stage breaks down the sentence into its basic word components. Our refined sentence from the previous step would be tokenized into the following: ["I've", "recently", "started", "working", "with", "LCDP"].
- Stemming: This procedure simplifies words to their root form to ensure consistency across various usages. For instance, the word "working" from our tokenized list would be stemmed down to its root: "work".

In a next step, we use a normalization for the posts, which defines a relevance score to each post, which is being composed by scaling the number of *views*, *scores,* and *answers* between 0 and 1 (normalization).

3.2.2. Finding Optimal Number of Topics. Once the data is cleaned, we perform probabilistic topic modelling with the use of an LDA algorithm. A considerable part of this procedure is hyper-parameter tuning. Hyper-parameter tuning is an iterative process that aims to find the best definition of parameters for the specific model (Bardenet et al., 2013), which is a common process in software engineering research (Abdellatif et al., 2020; Arun et al., 2010; Bagherzadeh and Khatchadourian, 2019).

For this step, our objective is to determine the optimal number of topics K for our dataset B to ensure that the coherence score is high, i.e., underpinning topic encapsulation. For large data-sets a higher number of topics tends to work better, whereas smaller data-sets generate better results with a smaller number of topics (Hasan et al., 2021). We conducted hyper-parameterization on a subset and optimized it for the coherence score. Following previous work (Röder et al., 2015), we used the Gensim package (Rehurek & Sojka, 2010) to calculate the coherence score (which is described within section 3.3). We tested our subset with different values of K ranging from 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, and 70 and run it on our dataset for 1000 iterations for each value (Bagherzadeh & Khatchadourian, 2019). Then, we investigated how the coherence score changes in relation to K. During this phase we tested with how many topics and how many iterations the model performs most accurately. Thus, the topic model with the highest coherence score is chosen. In our case, this was with 10 topics. Once we found the optimal composition of the LDA model, we initiated the model and analyzed the returned topics.

3.2.3. Generating Topics. Topic modeling is a technique for extracting a set of topics from a collection of documents that lacks a predefined classification system. Each document has a topic probability distribution, and each topic has a set of related word probability distributions. Thus, the LDA model will return a list of the most important words for every topic as displayed in Table 1.

Table 1: Generated topics.

#	Topic	Keywords	Subject Areas
1	Graphical user interfaces	page, make, visualforce, button,display	Visualization
2	Creating, modifying and filtering tables	slicer, column, table, select	Database and table management
3	Characteristics of tables	date, calculate, month	Database and table management
4	API requests	API, rest, access	Third party integration
5	File handling	save, load, report, view, embed	Handling files
6	Local and interconnected database management	SQL, connect, database, server	Third party integration
7	Third-party program integration and support	third-party integration	Third party integration
8	Other aspects	pass, record, use	Other aspects
9	Database querying	query, type, function, soql	Database and table management

10	Visualization problems	show, field, chart, trigger	Visualization

3.2.4. Model Evaluation. The next question that arises, is how well the model performs. When it comes to LDA there are several evaluation methods. The aim of those is to give insights on the model's performance. For LDA models there are two metrics that are most often used. One of them being the coherence score and the other one being the perplexity score (O'callaghan et al., 2015). The coherence score is a metric that evaluates the degree of semantic similarity between relevant words in the topic (O'callaghan et al., 2015). The perplexity score on the other hand measures how surprised the model is when getting fed with new unseen data and is measured as the normalized log-likelihood of a held-out test set (Shashank, 2019). However, Shashank (2019) as well as O'callaghan et al. (2015) state that the optimizing for perplexity score oftentimes does not make sense as it does not yield human interpretable topics. Therefore, we chose to optimise for coherence score. A set of sentences are "coherent" if they support one another (Shashank, 2019). There exist six different ways how to measure coherence. Accordingly, to prior research (see Al Alamin, 2023), we use the Cv measure, which is based on a sliding window and uses an indirect confirmation measure that uses normalized pointwise mutual information as well as the cosine similarity. As a first step, we need to figure out, with which parametrization of the POS() method leads to the best coherence score. Therefore, we construct a for loop creating different data frames with different POS parameters. Three settings were compared: the first allowing all POS tags, n v regarding nouns and verbs and n a regarding nouns and adjectives. For each of those data frames, the LDA algorithm was applied three times. Because of computing reasons, all model tuning steps are applied on a subset of the data, containing 3000 randomly assigned posts. Since we do not know the correct parametrization of the number of topics and the number of iterations, we will always train the model for 7 topics and 500 iterations. This decision stems from various test-cases indicating a fairly nice coherence score with these parameters. Afterwards, we could compare the resulting average coherence score for each of the POS parametrizations resulting in: 0.5178 for all words, 0.516 for nouns and adjectives and 0.530 for nouns and verbs. The coherence score is thus maximized when looking only at nouns and verbs. As in our research context sentiments are not relevant, this makes sense since adjectives could create noise.

Next we had to decide, whether we want apply the LDA model based on the "body" or the "title" attribute of each post. Arguably, the informational value is more condensed in the title column. On the other hand, the "body" attribute of a post might contain more detailed information. Again we compare the two settings relying on the coherence score as a performance indicator. On top of that we manually interpret the results of the model. To do so, we apply the LDA algorithm to both settings and compare the coherence score. On top of that, the result of the algorithm was manually inspected. The resulting coherence scores were 0.5133 for "Body" and 0.3963 for "Title".

4. Findings

The following section will illuminate the key trends and patterns observed in our data set and subsequent analysis. We will subsequently outline potential solutions to the challenges that developers commonly face.

4.1. Descriptive Results of Data Set

Initially, we conducted a descriptive analysis of our data set to better understand the time-frame and scope of the issues faced by developers. Figure 2 shows that low-code related posts on StackOverflow increased in the last years. This growth indicates increasing attention for LCDPs and thus justifies further research efforts in this area.

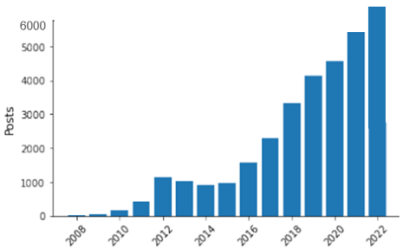

Figure 2. Number of posts per year.

Upon examining the distribution of posts across platforms, we observe a distinct emerging trend. Power BI has witnessed a significant surge in discussion accounting to almost 80% of posts in 2022. This contrasts with Salesforce and Progress, which both have seen a decline in their representation in SO posts; particularly Progress is scarcely mentioned in recent posts. Interestingly, Google-app-maker, which enjoyed popularity between 2017 and 2019, has since lost importance. However, interpreting those trends warrants caution, since it is ambiguous. An increase in posts for a platform could be attributed to its expanding user-base. Conversely, it might also

suggest heightened challenges faced by users, potentially stemming from software updates or other changes. Yet, irrespective of these interpretations, one observation is clear: Power BI's dominant presence in discussions.

By analyzing the number of views and answers as well as the score (likes - dislikes), we can make a statement about the relevance of a post. The data set has the following properties:

Table 2. Relevance metrics.

Column	Max	Mean	Median	SD	Weight
Views	125'713	1'223	274	3'647.09	0.5
Score	3	0.63	0	1.81	0.2
Answers	13	1.02	1.0	0.8	0.3
Relevance	0.81	0.0297	0.0253	0.0293	

In order to work with a definitive metric, we normalize each of the attributes between 0 and 1 and weigh the normalized values according to the" Weight". By doing so, we get an even deeper understanding of the relevance of the posts. Indeed, the data reveals some intriguing trends regarding user engagement on posts about LCDPs. It shows that more than 50% of all posts have exactly one answer. The vast majority of posts receive fewer than five responses, suggesting that posts with more than five answers are a rarity. In terms of post scores, a significant number register a score of 0, indicating limited user endorsement or recognition. Conversely, posts with scores exceeding 5 are a rarity. As for views, a prominent pattern emerges: over 5000 posts have attracted between 0 and 50 views, but as the view count rises, the number of corresponding posts decreases markedly.

These trends, when taken collectively, suggest an interesting dynamic: while individual posts might not always garner extensive engagement, the cumulative metrics underscore the escalating relevance of LCDPs in ongoing discussions. Further, in recent years, the number of answers grew, indicating a larger active community among the users of LCDPs. A peak of around 5.9 million total views becomes apparent in 2018. However, all metrics drop sharply from 2018 onwards. This could be an indicator that these platforms have lost their attractiveness or that the challenges faced by the community have become more difficult to solve and thus members are more reluctant in terms of answering questions. The fact that the total number of posts increased (see Figure 2) suggests that the latter is the case. Therefore, it is important to analyze the problems in detail.

4.2. Subject Areas

Initially, we identified 10 LCDPs-related themes. To achieve this, we first conducted a stratified sampling of the vast dataset of 30,000 StackOverflow posts. By selecting representative samples across various timeframes and topics, we ensured that we captured the essence of the broader dataset. After this sampling, our team manually reviewed and labeled these selected posts. These themes covered a broad range of topics and encapsulated the diverse problems developers encounter while using LCDPs. To better comprehend the nature of the challenges, we sought to consolidate these themes into high-level categories. This process involved examining the core issues within each theme and identifying commonalities or overlaps with other themes. For example, themes that addressed user interface problems or visualization challenges were grouped under the "Visualization" category.

After careful deliberation, we grouped the initial 10 themes into five high-level categories: (1) Visualizations, (2) Database and table management, (3) Third party integration, (4) Handling files, and (5) Other aspects. Each category represents a broad area of concerns related to LCDPs and encapsulates several of the initial themes.

Each category and its constituting themes were then revisited and evaluated to ensure a logical, meaningful consolidation that accurately represents the issues encountered by developers when using LCDPs.

The subject area of "***Visualization***" contains questions regarding User-Interfaces (94.3%) and displaying tables (5.7%). 20% of the questions concern front-end problems, 42.9% back-end problems and 31.4% are about problems that arise when connecting the front and the back end. Programmers tend to have problems visualizing objects in visualforce and also updating the UI's after the user made an action. The category "***Database and table management***" consists of two sub-fields and covers the area of creating and modifying tables (90.5%) and databases (9.5%). When it comes to filtering and querying (42.9%) tables with boolean expressions, a typical question would be "How to filter my sales table for products including a discount". These questions tend not to be too complex and rather repetitive. This is similar for the table creation and modification (28.6%), which includes general questions regarding how to establish tables. Lastly, calculations (19.0%) include basic numerical calculations with tables. An example is how to add a column containing the total of the previous two columns. Most posts in this area concern getting the

total amount across rows and columns. The complexity of this subcategory is rather low again.

The second sub-field is especially concerned with database creation. This includes entity relationship and normalization problems. A representative question is ”How can I create a database in Zoho?”. We note that the complexity of these questions is higher compared to the table-related questions.

The area “***Third-party integration***” covers API-related questions (Requests 20.7% and authentication 31.0%) and regards the migration to other platforms and services. API-related questions can be divided into request-related and authentication-related questions. While request-related questions mostly concern best practices on how to handle certain responses from API. Most questions in this subcategory are related to the Salesforce API. Authentication-related questions, on the other hand, are mostly asked by developers having problems sending the correct parameters and tokens when requesting access to a certain API. Again, we note that these questions have a high degree of complexity. Lastly, third-party integration regarding other platforms and services includes the automated distribution of emails (8.6%) as well as different migration problems (39.7%). Migration problems occur whenever one tries to connect two or more individually working implementations together. Typically, these questions indicate that users have two separate, functional environments, but they encounter issues when attempting to integrate or migrate between the two. .

(71.1%) includes questions that address problems when switching from another programming language to a platform. In this context, our analysis indicates that Python, Java, and JavaScript were mostly mentioned. The second sub-category includes comparison (28.9%) which consists of questions people asking which LCDP is best suited to do certain tasks. A typical question would be” Which platforms is suited best to populate a customer’s database.

Next, we analyzed the relationship between the subject areas and the LCDPs’ providers and if one of the LCDPs is disproportionately mentioned. Within our analysis we see that “***Database and table management***” seems to be tied to Power BI since Power BI has an over-representation of 45.7%. The over-representation simply comes from the following consideration. Power BI is associated with 51.8% of all posts whilst being associated with 75.4% of the posts that are assigned to this category. It also becomes apparent that problems related to this category occur below average for Salesforce, UI path, and Google-app-maker. Further, we can see that ”***Third-party integration***” problems are strongly tied. We note that Google-app-maker is mentioned surpassingly within the said subset. Additionally, we can deduce that problems within this subject are less likely to appear with Power BI. Considering the ”***Visualization***” category, we can clearly see that it seems to be tied to Salesforce as it is overrepresented by 63.4%. Along it becomes apparent that these problems are less important for Power BI and Google-app-maker. It

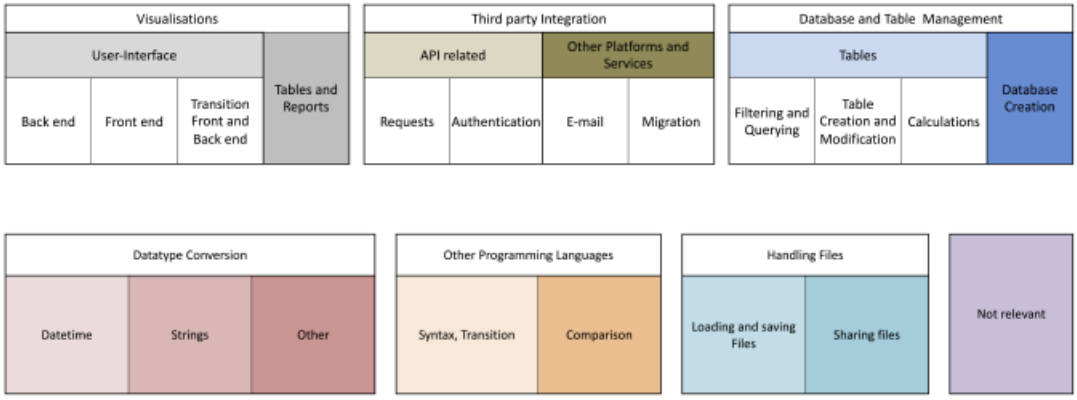

Figure 3. Overview of initial topics.

The fourth category, “***Handling Files***” is concerned with two major problems. One would be the loading and saving of files (50.0%). This problem arises whenever programmers want to import or save local files via LCDPs. The second issue considers upcoming problems when sharing files (50.0%) via LCDPs.

Fifth, the area of “***Other aspects***” is concerned with syntax and comparison. The subarea syntax

follows that this is an issue for Salesforce especially. Looking at the fourth graph, we can see that Zoho and UI path are overrepresented by 88.6%, respectively 59.4%, indicating that connecting to other programming languages seems to be a major problem in Zoho and UI path. Lastly, our investigation indicates that ”***File handling***” problems occur above average in Power BI and Zoho whilst being less problematic for Google-app-maker. In summary, it

can be stated that Power BI has problems regarding "***Database and table management***" whilst Salesforce is above average mentioned in the context of "***Visualization***". Google-app-maker is often mentioned regarding "***Third-party integration***" while problems concerning "***Other aspects***" appear above-average in Zoho and UI path.

5. Discussion

Our analysis unveiled novel insights into the challenges faced by users of LCDPs. The overarching concern revolves around creating, filtering, and modifying tables, followed by issues with third-party integration and visualization. Our findings, however, go beyond mere identification of these problems and have profound implications for the understanding of LCDPs and their effective utilization. In essence, we have highlighted the nuances of the issues that arise within each subject area, providing direction for addressing these concerns. We observed that most questions pertaining to creating and modifying tables are relatively non-complex and can be readily addressed with step-by-step instructions or video tutorials. This finding not only underscores the educational needs of LCDP users but also highlights an area for LCDPs to improve user guidance, thereby enhancing their user experience and efficacy.

Whilst manually labelling the posts became apparent that the most relevant questions in this subject area are non-complex and easy to solve problems. We therefore suggest that LCDP's introduce step-by-step instructions or video-tutorials covering the most basic techniques of creating and modifying tables. In case there is a need for prioritization, they should first focus on filtering, then on table creation, and lastly on basic calculations. It showed that problems concerning "***Database and table management***" are mostly tied to Power BI. Additionally, the manual labeling part revealed problems with converting date-time row entries into other formats.

Given that "***Third-party integration***" is the second most prominent subject area over all and in 2022 our next proposition is to address these problems as a second priority. The first subarea consists of problems regarding accessing and authorizing API's. Manually labelling these posts revealed the complexity of them. However, problems with the basic authorization requests repeated often. Therefore, we advise LCDPs to publish concrete examples with instructions on how to access API's from or through LCDPs. The most prominent subarea of "***Third-party integration***" concerns the migration to other platforms and services. The manual labelling part showed that these questions are very diverse. Therefore, we advise LCDPs to extend customer support and specifically hire professionals who answer these questions individually. Such a service would satisfy the needs of developers. When it comes to the "***Visualization***", this subject area mostly covers UI related problems. In case there is a need for prioritization, LCDPs should first focus on problems dealing with the connection from front and back-end, then focus on front-end problems and lastly on back-end problems. Since the term "visualforce" appeared in 6.64% of the posts in this subject area, LCDP's should especially enhance the connectivity to that platform. Our analysis shows that "***Visualization***" related problems are especially important to address for Salesforce. Based on our findings we suggest LCDP's to address the "***Visualization***" related questions by uploading tutorials explaining the basic concepts of front and back-end handling. It has also been shown that the most common issues arise in connection with python, java and Javascript. Therefore, we recommend that LCDP's focus more on the development of interfaces with these three programming languages. In the manual labelling process, it became clear that in about one third of the posts in this topic area people asked which LCDP is best suited for a particular task. As these are again individual concerns, LCDPs are encouraged to educate their customer service staff on the pros and cons of their respective platform. Even though "***File handling***" only represents 6.4% of all posts in 2022, it showed a very stable development throughout the observed period. Manual labelling revealed that the questions in this subject area are mostly very simple and repetitive. Furthermore, we assume that these problems will always exist, regardless of the development of technology, as almost every program will contain some sort of loading and storing files.

Our research is not without limitations. First, despite having many observations, the dataset is still somehow limited, since LCDPs are a novel technology. Furthermore, we only looked at StackOverflow posts. Having in mind that citizen developers might act differently from professional developers, it remains unclear, if they look for help on StackOverflow and if they share their knowledge in this kind of online community. Future research could combine this LDA approach with an interview study and thus clarify where especially citizen developers would post their issues. Thirdly, we had a holistic view on all of the LCDPs. Thus, another limitation arises from the data sources used. Because many LCDP providers have their own forums, some LCDPs may not have a lot of useful discourses on StackOverflow. Richer insights might result from analyzing posts for

one particular LCDP, for instance, Mendix. Lastly, we did not take any sentiments into consideration. Therefore, in the future, more aspects of the posts could be addressed. Our study has opened the avenue for several potential areas of future research. While our focus was on analyzing StackOverflow posts, it would be intriguing to explore other online forums or communities where LCDP discussions occur. Citizen developers' behavior and preferences could be analyzed more deeply, possibly by comparing different platforms or geographies. Additionally, a sentiment analysis of the posts could reveal the intensity of challenges faced by developers, providing deeper insights into the severity of specific issues. A comparative analysis between different LCDPs might also offer a more granular understanding of individual platform strengths and weaknesses.

6. Conclusion

Recent years have seen an increasing uptake of low code development platforms in organizations. Especially factors such as increasing affinity for technology development across all user groups, consumerization of development, and advancing digitalization are opening a new target group of developers for the low code movement. Within companies, low code development platforms are a novel paradigm for developing applications with the minimal effort needed. In this paper, we present an empirical large-scale study by specifically investigating LCDPs-related questions on StackOverflow. We extracted LCDP-related posts from StackOverflow and then used advanced topic modelling to cluster different issues that developers face when developing with LCDPs. After obtaining different topics, we used the gathered metadata to investigate the dataset descriptively. We found that LCDP-related issues and questions on StackOverflow cover a great variety of topics in more than 30’000 posts. With our research we contribute to practice by uncovering what developers, citizen or professional software engineers, consider as the most challenging by shedding light on how those “how to”-type questions could be addressed. Further, we contribute to the ongoing debate about the challenges and issues with LCDPs. Thus, we contribute to this emerging literature stream.

7. References

Abdellatif, A., Costa, D., Badran, K., Abdalkareem, R., & Shihab, E. (2020). Challenges in chatbot development: A study of stack overflow posts. Proceedings of the 17th International Conference on Mining Software Repositories, 174–185.

Adrian, B., Hinrichsen, S., & Nikolenko, A. (2020). App development via low-code programming as part of modern industrial engineering education. International Conference on Applied Human Factors and Ergonomics, 45–51.

Al Alamin, M. A., Malakar, S., Uddin, G., Afroz, S., Haider, T. B., & Iqbal, A. (2021). An empirical study of developer discussions on low-code software development challenges. 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR), 46–57.

Alamin, M. A. A., Uddin, G., Malakar, S., Afroz, S., Haider, T., & Iqbal, A. (2023). Developer discussion topics on the adoption and barriers of low code software development platforms. Empirical Software Engineering, 28(1), 1–59.

Albon, C. (2018). Machine learning with python cookbook: Practical solutions from preprocessing to deep learning. O’Reilly Media, Inc.

Almonte, L., Cantador, I., Guerra, E., & de Lara, J. (2020). Towards automating the construction of recommender systems for low-code development platforms. Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, 1–10.

Arun, R., Suresh, V., Veni Madhavan, C. E., & Murthy, N. (2010). On finding the natural number of topics with latent dirichlet allocation: Some observations. Pacific-Asia Conference on Knowledge Discovery and Data Mining, 391–402.

Asmussen, C. B., & Møller, C. (2019). Smart literature review: A practical topic modelling approach to exploratory literature review. Journal of Big Data, 6(1), 1–18.

Bagherzadeh, M., & Khatchadourian, R. (2019). Going big: A large-scale study on what big data developers ask. Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 432–442.

Bardenet, R., Brendel, M., Kégl, B., & Sebag, M. (2013). Collaborative hyperparameter tuning. International Conference on Machine Learning, 199–207.

Baumgarten, C., Simeon, A., & Wilhelm, M. C. (2020). Citizen Developers Driving the Digital Campus. European Journal of Higher Education IT, 1.

Bird, S. (2006). NLTK: The natural language toolkit. Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions, 69–72.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. Journal of Machine Learning Research, 3(Jan), 993–1022.

Bock, A. C., & Frank, U. (2021). Low-Code Platform. Business & Information Systems Engineering, 63(6), 733–740. https://doi.org/10.1007/s12599-021-00726-8

Brinker, S. (2018, May 29). Democratizing martech: Distributing power from IT to marketing technologists to everyone. Chief Marketing Technologist. h

Danhieux, P. (2022). Council Post: Navigating The Developer Shortage Crisis: A Time To Define The Developer Of The Future. Forbes.

Daniel, G., Cabot, J., Deruelle, L., & Derras, M. (2020). Xatkit: A multimodal low-code chatbot development framework. IEEE Access, 8, 15332–15346.

Di Sipio, C., Di Ruscio, D., & Nguyen, P. T. (2020). Democratizing the development of recommender systems by means of low-code platforms. Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, 1–9.

Elshan, E., Dickhaut, E., & Ebel, P. (2023). An Investigation of Why Low Code Platforms Provide Answers and New Challenges. HICSS.

Frank, U., Maier, P., & Bock, A. (2021). Low code platforms: Promises, concepts and prospects. A comparative study of ten systems. In ICB Research Reports (No. 70; ICB Research Reports). University Duisburg-Essen, Institute for Computer Science and Business Information Systems (ICB).

Fryling, M. (2019). Low code app development. Journal of Computing Sciences in Colleges, 34(6), 119–119.

Hasan, M., Rahman, A., Karim, M., Khan, M., Islam, S., & Islam, M. (2021). Normalized approach to find optimal number of topics in Latent Dirichlet Allocation (LDA). Proceedings of International Conference on Trends in Computational and Cognitive Engineering, 341–354.

Jelodar, H., Wang, Y., Yuan, C., Feng, X., Jiang, X., Li, Y., & Zhao, L. (2019). Latent Dirichlet allocation (LDA) and topic modeling: Models, applications, a survey. Multimedia Tools and Applications, 78(11), 15169–15211.

Kedziora, D. (2022) Botsourcing, Roboshoring or Virtual Backoffice? Perspectives on Implementing Robotic Process Automation (RPA) and Artificial Intelligence (AI), Human Technology, Vol. 18, No. 2, pp. 92-97

Khorram, F., Mottu, J.-M., & Sunyé, G. (2020). Challenges & opportunities in low-code testing. Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, 1–10.

Kletti, N.-L. (2021). Trends der Fertigungs-IT 2021. Zeitschrift Für Wirtschaftlichen Fabrikbetrieb, 116(5), 276–278.

Luo, Y., Liang, P., Wang, C., Shahin, M., & Zhan, J. (2021). Characteristics and Challenges of Low-Code Development: The Practitioners' Perspective. Proceedings of the 15th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), 1–11.

O'callaghan, D., Greene, D., Carthy, J., & Cunningham, P. (2015). An analysis of the coherence of descriptors in topic modeling. Expert Systems with Applications, 42(13), 5645–5657.

Pantelimon, S.-G., Rogojanu, T., Braileanu, A., Stanciu, V.-D., & Dobre, C. (2019). Towards a seamless integration of iot devices with iot platforms using a low-code approach. 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), 566–571.

Prinz, N., Huber, M., Riedinger, C., & Rentrop, C. (2022). Two Perspectives of Low-Code Development Platform Challenges – An Exploratory Study. PACIS 2022 Proceedings. https://aisel.aisnet.org/pacis2022/235

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.

Röder, M., Both, A., & Hinneburg, A. (2015). Exploring the space of topic coherence measures. Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, 399–408.

Rymer, J. R., Koplowitz, R., Leaders, S. A., Mendix, K., are Leaders, S., ServiceNow, G., Performers, S., MatsSoft, W., & are Contenders, T. (2019). The Forrester WaveTM: Low-Code Development Platforms For AD&D Professionals, Q1 2019. Forrester Research.

Sahay, A., Indamutsa, A., Di Ruscio, D., & Pierantonio, A. (2020). Supporting the understanding and comparison of low-code development platforms. 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 171–178.

Sanchis, R., García-Perales, Ó., Fraile, F., & Poler, R. (2019). Low-code as enabler of digital transformation in manufacturing industry. Applied Sciences, 10(1), 12.

Shashank, K. (2019). Evaluate Topic Models: Latent Dirichlet Allocation (LDA) | by Shashank Kapadia | Towards Data Science. https://towardsdatascience.com/evaluate-topic-model-in-python-latent-dirichlet-allocation-lda-7d57484bb5d0

StackOverflow. (2022). Newest Questions. Stack Overflow. https://stackoverflow.com/questions/

Sun, S., Luo, C., & Chen, J. (2017). A review of natural language processing techniques for opinion mining systems. Information Fusion, 36, 10–25.

Tisi, M., Mottu, J.-M., Kolovos, D. S., De Lara, J., Guerra, E. M., Di Ruscio, D., Pierantonio, A., & Wimmer, M. (2019). Lowcomote: Training the next generation of experts in scalable low-code engineering platforms. STAF 2019 Co-Located Events Joint Proceedings.

Torres, C. (2018). Demand for programmers hits full boil as us job market simmers. Bloomberg.

Ugur. (2021, August 19). How many Low-Code/No-Code platforms are out there? SpreadsheetWeb. https://www.spreadsheetweb.com/how-many-low-code-no-code-platforms-are-out-there/

Vincent, P., Iijima, K., Driver, M., Wong, J., & Natis, Y. (2019). Magic quadrant for enterprise low-code application platforms. Gartner Report.

Waszkowski, R. (2019). Low-code platform for automating business processes in manufacturing. IFAC-PapersOnLine, 52(10), 376–381.

Wessel, L., Baiyere, A., Ologeanu-Taddei, R., Cha, J., & Blegind-Jensen, T. (2021). Unpacking the difference between digital transformation and IT-enabled organizational transformation. Journal of the Association for Information Systems, 22(1), 102–129.

Yang, X.-L., Lo, D., Xia, X., Wan, Z.-Y., & Sun, J.-L. (2016). What security questions do developers ask? A large-scale study of stack overflow posts. Journal of Computer Science and Technology, 31(5), 910–924.