How little code is low-code? - Towards productivity measures for the use of low-code development platforms by business user developers

Olga Levina
Department of Business Management
Brandenburg University of Applied Sciences
Brandenburg an der Havel, Germany
email: olga.levina@th-brandenburg.de

Katharina Frosch
Department of Business Management
Brandenburg University of Applied Sciences
Brandenburg an der Havel, Germany
email: frosch@th-brandenburg.de

***Abstract*— Low code development platforms (LCDP) often promise an easy and fast way to include data processing and support into the otherwise non-digital process. This research explores how to measure the productivity of low code development to assess the effort needed for business users to respond to their need for support via these tools. We chose field experiments as a research method to evaluate the feasibility and derive the metrics for software development with LCDP by novices. The paper provides some insights on how these measures can be implemented in practice, how to support business unit developers to efficiently deliver productive results, and how to evaluate LCDP-based development processes.**

Keywords- low code development platforms; software development process; digital novices, productivity; performance indicators

I. INTRODUCTION

Demand for data management solutions in a business context, coupled with the challenge of modernizing legacy systems is fueling the innovation of new software development tools and methods. To create an application or be productive in manual coding, the programmers need to be skilled in specific programming languages. As skilled IT staff is scarce, this development creates a positive environment for the adoption of Low Code Development Platforms (LCDPs). This paper builds on the findings by [1] in the context of Business User Development of business applications using LCDPs. While this previous research explored the suitability of LCDPs to answer the data management needs of business users and the platform's potential to provide them with a satisfactory development tool turning business users into Business User Developers, the expansion of this research focuses on the determination of the productivity of the LCDP use in a specific business software development project.

LCDPs promise an easy and fast possibility to include data processing and support in the otherwise non-digital process [2]. The terms "citizen developer" or "Business Unit Developer" (BUD) [3] are often used in the LCDP context to underline the potential of the software tools to involve programming novices in the development of solutions for their needs [4].

Low-code platforms abstain as far as possible from using textual programming that requires manual coding and offer instead visual or, less often, natural languages [5]. As a result, developing applications using low-code technologies is faster and may result in swifter delivery and higher productivity [6]. Thus, this research addresses the following research questions: How can the effort needed to create an application with an LCDP by BUDs be assessed? As well as, how can the effort for software development using a programming language versus the development of the same requirements using LCDPs be compared?

As LCDPs were shown by [1] to be a usable tool for novices to address their digitalization needs, this research expands this question and enriches the usage and implementation of LCDPs by providing indicators for effort assessment in the context of software development projects.

In particular, we suggest a metric for the evaluation of LCDPs in terms of programming effort – the Low Code Factor (LCF), which is defined as the number of actions taken by the developers on the LCDP per use case. This metric will allow an assessment of LCDPs in terms of their effectiveness in fulfilling the digitization needs of the BUDs. It is based on UCPA (Use Case Points Analysis) [7], the effort assessment method for object-oriented software development projects, which we extend by the user interaction data with the LCDP.

As the research method, we use an experimental setting, where software application requirements are derived and documented by BUDs. Then we let BUDs create applications using an open source LCDP Joget. Based on LCDP activity logs, we evaluate the effort invested by BUDs to develop an application with an LCDP. As a novel contribution, we suggest LCF as a measure of BUD's productivity on the LCDP. A further metric, LCDPfit aims to provide project managers with means for the assessment of the project size and effort needed to complete the project. Also, a cost-benefit calculation of the planned software

realization using the two different approaches (low code vs. classic software development) can now be achieved.

The paper is structured as follows: First, we review the current literature on how LCDPs are currently used in a business context, and what methods are commonly used for productivity assessment in software development. Then we derive productivity measures, in particular the LCF, that we then apply in our experimental setting. The results obtained from the analysis lead to recommendations for implementing LCDPs in a productive environment. We close with a summary and outlook on future research.

II. Related Work

A. Use of LCDP

The use of LCDPs in different business domains has been increasingly the focus of research in the last few years. Sanchis et al. [8] showed that rapidity and cost reduction through intuitive development and management can be attributed to the use of an LCDP in a manufacturing context. Nowak et al. [9] showcase the usage of LCDPs in the context of the internal logistics processes in a company from the E-Commerce industry. This case study is meant to display the use of LCDPs in the context of process improvement as it allows for the direct elimination of found limitations in processes. The authors argue that the implementation of the IT support using LCDPs was effective, i.e., an enhancement in terms of time and costs needed for its realization.

Bies et al. [10] conducted a mixed-method study to identify challenges and promising perspectives for digital innovations in small and medium-sized enterprises (SMEs). The authors found that the application areas of LCDPs are mostly of a supportive nature such as the creation of applications for resource management or the creation of customized digital forms. Nevertheless, the majority of the surveyed SMEs stated LCDPs to be of high to very high relevance. Factors that diminish the relevance of low code in SMEs are according to the authors: limited human resources, as personnel is still necessary to develop and maintain the application, knowledge transfer between the platforms as well as training in dealing with IT structures and detailed knowledge of the platforms.

Lethbridge [11] also explores the development process of the software product as well as the aspects of implementation and maintenance of the LCDP software within the existing enterprise architecture. His findings suggest that LCDPs create “technical debts” that can be overcome by the development of the LCDP towards “scaling, understandability, documentarily, usability, vendor- independence and user experience for the developers”. Hintsch et al. 2021 [12] also identify threats and opportunities in the LCDP development concerning the security and availability of the created applications. Nevertheless, the authors also uncover success factors for LCDP use in a business context by novices.

Kermanchi et al. [13] focus in their research on software development methods and the use of LCDPs. In their experiment, they explored the episodic experience with different LCDPs among software developers with varying levels of programming experience but no experience in the specific LCDP. The findings show that previous programming experience seems to have a significant impact on developers' performance, experiences, and tool preferences, yet most developers continue to have doubts about the scalability and maintainability of applications created with LCDPs. The opinions on the effectiveness of the instruments vary among the participants.

Bernsteiner et al. [14] conduct expert interviews in their research to investigate what skills developers with little or no software development experience, i.e., novices, need to successfully develop software on LCDPs. Several of the interviewed experts mention that successfully developing an LCDP solution requires at least basic programming skills. This is in line with research findings stating that LCDPs still require some prerequisites in software development [15] or in database structures [16], which hampers the adoption of LCDPs by non-programmers without any further training.

Krejci et al. [16] report in a case study how non-IT employees were involved in the process of digital innovation while making efficient use of their IT resources. These citizen developers, i.e., employees who are working outside of the Information Technology (IT) department and are not professional programmers, as users of LCDPs are the focus of the analysis by Lebens et al. [17]. The authors surveyed the use of LCDPs in organizations. The results show that companies both large and small are making use of low- and no-code platforms. Additionally, the majority of the surveyed organizations have employees outside of the IT department who are creating IT solutions.

Bock and Frank [18] provide a critical overview of the LCPD features, architecture, and opportunities while pointing out research directions for information systems research in this domain. They state that although both professional developers and citizen developers use LCDPs, there is a lack of research on how to make LCDPs fit the cognitive capabilities and personal working styles of these two groups [p. 739]. This is in line with other studies pointing out that successfully developing software on LCDPs requires at least basic programming skills.

The use of development templates in the context of software creation is analyzed by Boot et al. [19]. The authors compare instructional software products made by developers with low production experience and high production experience, working with a template-based authoring tool. The analysis showed that the technical and authoring quality was equal for both groups, indicating that templates enable domain specialists to participate successfully in the production process. Research in agile software development shows that projects based on the Scrum methodology profit from having a coach on the team [20]. The same is visible in software engineering education [21].

BUDs and job crafting, i.e., proactive strategies to improve work processes according to one’s own needs and goals, are subjects of the analysis by Li et al. [3]. The authors found that using LCDPs provides positive job

crafting consequences such as meaningfulness, for the employees using these tools [3], [22]. In what follows, we prefer to use the term BUDs instead of citizen developers, stressing that they might make up for the lack of programming skills with their large expertise in the respective business domain. Nevertheless, the research does not focus on the description of how much support was needed for BUDs to finish their application.

In conclusion, in these first attempts to understand the "human side" of LCDPs, research is still scarce concerning acceptance and successful adoption by domain experts outside corporate IT departments. We also lack information on how effective (or productive) BUDs are in using LCDPs to fulfill their own digital business needs.

B. Productivity assessment in software development

How widely LCDPs will be used in enterprise context by BUDs without sound programming expertise might also depend on the productivity they can achieve with the respective tool.

The concept of productivity in software development is not new to the domain and has been studied from various perspectives. However, there is still no consensus within academic and industry circles, as some researchers argue that using a single metric to measure productivity can lead to problematic and misleading [6] assessments.

Hence, among the methods to assess productivity in software development are lines of code [23]; function point analysis [24]; and Use Case Points Analysis (UCPA). UCPA was developed in the context of object-oriented software development by Gustav Karner (see e.g., [7], [25]) and is similar to the function point analysis. We chose UCPA in our study following [6] as it provides a way to estimate the size and complexity of a software development project early on, based solely on requirements [26]. UCPA leverages use cases representing functional requirements as its starting point. The resulting Use Case Points (UCP) metric reflects the complexity of the project across three dimensions - functional, technical, and environmental, i.e., considering the context of the project. Thus, UCPA allows sizing and estimation of the effort required for a software development project.

Hence, to assess productivity in a software development project, we lend the definition from the economics discipline and define software development productivity simply as the ratio of outputs produced to the inputs involved in that production, also following [6], [27], [28] who use this definition in the software development domain. In the context of application development, input is defined as the time and activities invested in the development and the output will be the implemented use cases.

While UCPA presents a good tool for manual programming effort assessment, it does not account for the potential that the LCDPs are providing for the development project.

Given the research activities in the areas of LCDP usage in the business context, especially among BUDs, as well as the nature of finding a digital solution to a business problem being a software development project, the following research questions are identified:

- RQ1: How can the effort needed to create an application with LCDPs by BUDs be assessed?
- RQ2: How can the effort for software development using a programming language versus the development of the same requirements using an LCDP be compared?

III. RESEARCH METHOD

To answer the research questions, experiments were set up with the Master's students of Business Management and Information Systems. The goal of the experiments was to assess the effectiveness of the app development using the LCDP Joget, which is described further in [1]. Therefore, different scenarios requiring digital support were suggested for the students for their implementation in the app, using the LCDP. Based on the application design that was documented in activity diagrams, user stories, and mockups, as well as based on data logs from the experiment, the productivity metric was derived.

A. Data collection based on field experiments

To gain evidence for answering our research questions, we draw upon a field experiment where BUDs with little prerequisites in software development build app prototypes in the business domain of human resource management (HRM) based on an LCDP given a finite time frame of a few weeks. Overall, 13 HR apps have been developed.

The LCDP used for the experiment was Joget [29], an open-source LCDP with the promise to easily build, run, and maintain apps. A visual builder allows drag-and-drop for pages, forms, views, data lists, menus, and a process builder to automate workflows. It also offers user management and role-based authentication. We used the community edition that can be self-hosted at no license cost.

BUDs were Master's students of business management with a specialization in human resources management (HR) and Master's students of information systems management (ISM). All of the ISM students had already taken at least one course in advanced software engineering within their Master's program at the time of the experiment but were far from being experienced software developers. The HR students had no previous expertise in software development. None of the participants in either group was familiar with or had heard of the LCDP selected for the experiment. Figure 1 presents the data collection process and the sequence of the experiments.

The experiment was divided into four self-contained challenges with modified compositions of participants. The challenges are described below.

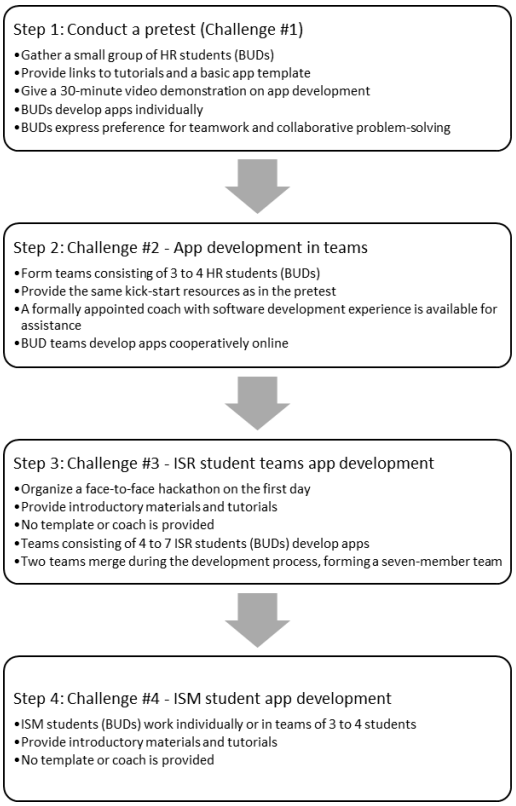

Figure 1. Process of the data collection in the field experiments

For the first two challenges, BUDs are Master's students of business management with a specialization in HR (six and 16 students, respectively). They sketched their app concept and subsequently implemented it. In the third and the fourth challenge master students of information systems management (ISM) were provided with ready-prepared HR app concepts and then asked to implement them (24 and eight ISM students, respectively). The apps were from the HR domain but otherwise differed in their content and scope.

Challenge #1 was run with a few HR students as BUDs (six) only, to have a pretest and check whether they are, at all, able to use the LCDP to develop simple apps. The pretest was run between April 21 and June 6, 2021 (47 days). To kick start app development BUDs were provided with links to tutorials as well as with a basic app template and a 30-minute video showing exemplarily how an app can be built starting from this template. In this context, they were also explicitly pointed to the open-source character of app development in this setting, and about the possibility to share and reuse app elements from other groups. In the pretest, BUDs managed to develop apps but pointed out that they would have enjoyed working in teams to solve problems collaboratively. Furthermore, support from one student who previously had graduated from a Bachelor's program in software engineering and acted as an informal coach for his fellow students has been acknowledged as extremely helpful.

Based on the insights gained in the pretest, we recruited the informal coach from challenge #1 to act as a formally appointed coach in challenge #2 and decided to run development in teams. For challenge #2 BUD teams (with three to four HR students, 16 in total) developed their apps within six weeks between November 1 and December 12, 2021 (42 days). The team members cooperated online, due to the restrictions because of the COVID-19 pandemic. Developers got the same kick start as in the pretest and were also pointed towards the template and the possibility to share and reuse apps. Furthermore, a coach with experience in software development was available to get help with questions on tool usage and minor development questions.

In Challenge #3, 24 BUDs in teams of four to seven ISR students developed their apps between May 20 and June 7, 2022 (19 days). The first day of the development phase (May 20, 2022) was organized as a face-to-face daylong hackathon. The introductory video and tutorials were made available beforehand, but no template or coach was provided for the teams. During the development challenge, two teams joined forces within the development process, resulting in a seven-member team working on the challenge.

Challenge #4 was a replica of challenge #3 with 8 ISM students acting as BUDs, where one worked alone and the others in teams of three or four students between May 24 and June 28, 2022 (36 days).

The effectiveness of using the LCDP to solve the business needs for BUDs was described in [1]. In this research, the focus is on the description of the productivity metric for the project assessment as well as for the assessment of the suitability of the LCDP for solving business-related questions compared to the software development using a programming language.

B. Measuring the coding effort on the LCDP

To evaluate efforts made by BUDs to develop their app, the LCDP activity logs were archived and anonymized. These data were used to calculate indicators to measure the effort invested in app development based on the LCDP. Note that there is no log data available for challenge #1. We use the following indicators related to time spent on the platform and the number of actions:

- *Time on the platform (hours)*: Total time a developer was active on the LCDP during the developing stage. Based on the first and the last action performed for each login identified, we can compute the duration users are active per login. Idle periods of 30 minutes or longer are omitted, assuming that the user then has stopped developing. Summing up yields the total time on the platform in hours.
- *Time investment (hours)*: Total time invested by app is obtained by summing up hours spent on the platform across all members of the developer team of the respective app.

- *Number of actions, by developer*: For this variable, we count actions taken by each developer, such as creating, editing, and deleting code, forms, views, or other assets.
- *Number of actions, by app*: Aggregation of actions undertaken by all members of the development team of the respective app.

As the duration of the development phase and team size vary across challenges, app-based indicators for effort invested are more informative as compared to effort indicators at the level of individual developers.

Using these indicators and relying on the methods for productivity measurement in software engineering described in section II, the LCDP-related productivity factors *LCF* and *LCDPfit* were derived and calculated.

C. The Low Code Factor (LCF)

The calculation of the *LCF* is based on the UCPA method. This method considers users involved in the interaction with the software as well as the interaction patterns, i.e., use cases of these actors. The UCPA method consists of several stages, see e.g., [25]:

First, the actors (roles interacting with the system) and use cases need to be identified. Then, the actors need to be classified into one of three categories based on the complexity of interaction with the system according to [25]:

- Simple actor- e.g., system interface, weight 1
- Average actor- e.g., protocol-driven interface, weight 2
- Complex actor- e.g., GUI, weight 3

Then, each use case needs to be classified based on its integration complexity as simple, average, or complex. Complexity assessment is based on aspects such as transactions, i.e., communication, information exchange, or data access, etc.

- Integrated use cases: already implemented transactions in the LCDP, weight 0
- Simple use cases: 1-3 transactions, <5 classes, weight 5
- Average use case: 4-7 transactions, 6-10 classes weight 10
- Complex use case: >8 transactions, >11 classes, weight 15

As LCDPs already provide some implemented interaction patterns, we suggest a new class of use cases that is specific to the use of LCDPs: the *integrated use case* with the weight 0, as no programming effort is required to implement this use case. Also, since the app development project was based on the LCDP-based development, the UML classes as referred to in UCPA were realized as "data lists" in Joget terms.

After the classification of the use cases, the productivity indicators need to be calculated (see [30] for calculation details):

- Unadjusted use case points (*UUCP*) are calculated as the sum of the unadjusted actor weight (*UAW*) and unadjusted use case weight (*UUCW*):
  - $UUCP = UAW + UUCW$
  - *UUCW* is calculated by multiplying the number of each use case type by a weighting factor according to its classification.

UAW is calculated by multiplying the number of actors by the weighting factor.

Now, UUCP needs to be adjusted using technical complexity factors (*TCF*) and environmental complexity factors (*ECF*) to derive adjusted use case points (*UCP*).

- The combination of the *UUCP* variable with the *TCP* and EF variables results in the actual number of *UCP* of the project:
  - $UCP = UUCP \times TCF \times ECF$
- TCF is one of the factors applied to the estimated size of the software to account for technical considerations of the system. It is determined by assigning a score between 0 (factor is irrelevant) and 5 (factor is essential) to each of the 13 technical factors. This score is then multiplied by the defined weighted value for each factor (*TF*).
  - $TCF = 0.6 + (TF/100)$
- ECF is determined by assigning a score between 0 (no experience) and 5 (expert) to each of the 8 environmental factors. This score is then multiplied by the defined weighted value for each factor (*EF*):
  - $ECF = 1.4 + (-0.03 \times EF)$

This value is multiplied by the productivity factor (*PF*), which represents the number of hours required to develop each *UCP*:

- $Total\ Effort = UCP \times PF$.

Tables II and III provide the calculations for selected apps from challenge #3. In sum, the productivity assessment in our context considers *UCP* as the output measure and PF as the input measure. To assess the productivity of the LCDP-based app development, we introduce the *LCF* and *LCDPfit* metrics that are based on the platform log data that was generated per app.

The Low Code Factor (*LCF*) assesses the effort submitted versus the functional complexity required for the realization of the business solution that is calculated using UCPA. To calculate the *LCF* we derive the number of actions performed per app (see Table I) and divide them per weighted use cases *UCP*. Thus, it provides the measurement of the platform interaction needed to realize the use cases. *LCDPfit* is calculated as the quotient of the number of lines of code needed to realize the app despite using an LCDP and the UCP. Thus, the *LCDPfit* provides an assessment of the programming effort required despite using the LCDP,

while *LCF* assesses the effort of the platform interaction for the realization of the app.

Calculation and interpretation of *LCF* and *LCDPfit* for productivity assessment across the presented challenges are described in the following section.

IV. RESULTS

The experiment has shown that in all challenges, BUDs were able to create a software application using an LCDP in a given amount of time without any (challenges #1 and #2) or at least no extensive professional training (challenges #3 and #4) in software development, see also [1]. All apps created during the challenges have been successfully developed and implemented. "Successfully" means that they met the requirements depicted in the conceptual papers and that 13 apps worked when tested. The technology readiness of the prototypes corresponds to level 3 (experimental proof of concept) according to the European Union Technology Readiness Levels [31].

Overall, our data comprises 568 logins, resulting in 10,395 actions taken, respectively. The distribution of time spent on the platform is right-skewed, with most developers investing not more than 10 hours in development. Moreover, we observe two outliers with more than 60 (challenge #2) and more than 30 (challenge #3) hours, respectively. When analyzing effort at the level of developers, comparing means may lead to misleading results whereas modal values provide a more robust measure for typical development effort.

To gain more insights into what effort is needed to develop a business app using LCDP and analyze time spent on the platform and the number of actions taken by the app for each of the 13 apps that have been created across challenges #2 to #4 (Table 1).

TABLE I. EFFORT PER APP

App	Challenge	Total time	No. of actions
1	#2	20.28	929
2	#2	23.72	608
3	#2	81.41	2454
4	#2	25.13	807
5	#2	19.91	417
6	#3	30.91	951
7	#3	43.92	1344
8	#3	19.18	373
9	#3	30.03	946
10	#3	17.5	686
11	#4	10.92	363
12	#4	12.91	299
13	#4	14.51	207

Table 1 shows that the number of actions taken per app and time investment for development by app varies considerably. However, effort invested by the app does not necessarily seem to depend on previous programming expertise, as on average, the completely unexperienced BUDs in challenge #2 show a medium effort level concerning both, time and number of actions as compared to the somewhat experienced BUDs in challenges #3 (higher effort levels) and #4 (lower effort levels).

In the next step, we undertake productivity assessments for each of the 13 apps developed across challenges #2 to #4 using the suggested metric, the low code factor (*LCF*). This measurement will allow us to assess the effort submitted versus the functional complexity required for the realization of the business solution that is calculated using *UCPA*.

To assess the development productivity, *LCF* and *LCDPfit* are calculated. Table II shows the use cases and weights of the apps 6 –8 as well as their Technical Complexity Factor (*TCF*), Environmental Complexity Factor (*ECF*) as well as the productivity factor that is calculated as the quotient of the total effort (time spent on the app) and the weighted *UCP*.

TABLE II. UCPA CALCULATION OF THE APPS

App	No. of actors	Use Case	Weight
6	3	user login	2
		solve quiz	5
		view score	10
		view detailed score	10
		view feedback	10
		see score per applicant	10
		generate user	5
		manage questions	5
		manage evaluation guides	10
	UUCP	67	
7	3	login	5
		upload doc	5
		solve task	0
		view results	10
		view doc	5
		provide task	10
		check results	10
		send feedback	10
		CRUD results	15
		CRUD users;	15
		creates tasks	10
		solves tasks	5
	UUCP	100	
8	4	solve quiz	5
		view score	10
		view score per applicant	10
		generate evaluation	5
		manager users	10
	UUCP	40	

Table III shows the TCF and ECF of some of the apps as well the UCP according to the calculation of UUCP and adjusting it with the TCF and ECF:

- *UUCP = UAW + UUCP*
- *UCP = UUCP×TCF×ECF*

The productivity factor was calculated using the time spent per app from Table I and the *UCP* value.

TABLE III. METRIC OF THE APPS 6-8

App	*TCF*	*ECF*	*UCP*	*PF*	*LCF*	*LCDPfit*
6	1.02	1.1	84.85	0.36	11.21	9.09
7	1.02	1.1	123.36	0.36	10.90	3.55
8	1.02	1.1	58.06	0.33	6.42	5.34

Furthermore, Table III shows the *LCF* and the *LCDP*fit metrics for the selected apps. Assessment and data for all 13 apps are provided in the dataset at Zenodo [32].

The selected apps were designed by three different BUD teams according to the general requirements to build a mini assessment center for an HR responsible. Besides this general description, each team was supported by a "customer", i.e., an HR Master student who derived the requirements for the app and was supervising their implementation. All three teams did not have any previous knowledge of the LCDP in question, i.e., Joget, encountered similar values of the *TCF* and *ECF* in the UCP calculation. Despite similar basic conditions, the teams fulfilled their task with different functional extenuations. While app 7 realized twelve of the required use cases, team 8 realized five and team 6 nine use cases. Nevertheless, the teams showed similar productivity factors (see Table III). The efficiency of the LCDP use as indicated by the *LCF* and *LCDPfit* also varied between the teams, with team 6 engaging in the highest programming and LCDP engagement effort as shown by *LCDPfit* and *LCF* metrics respectively, and team 8 showed an efficient use of the platform and its given functionalities as shown by the *LCF*.

V. SUMMARY AND OUTLOOK

Using the results of the described experiment, we can draw the conclusion that BUDs can create their software applications in their business domain using an LCDP, and that time and effort invested in development are not significantly different between BUDs with no and BUDs with some programming knowledge. One interpretation of this result is that the LCDP used is really low code, as it does not seem to make a difference whether developers have no or some prerequisites in software development. Differences in the average effort displayed may for example result from individual performance preferences in the developer teams. Another possible explanation is that the complexity of the apps varied between challenges and also between apps within a challenge.

Besides the suitability of LCDP to support the realization of digitalization by BUDs this paper explored the possibility to measure the productivity of a software developer using LCDP as well as to provide an estimate for the effort needed to compose a business app using a LCDP. Therefore, an experiment with three different challenges was conducted. All the solutions for the challenges led to an app that was ready to be implemented in the business context. Although the quality of the created artifacts was not measured, and the size of the developer groups varied, the research offers valuable insights into the development process using LCDP by both non-IT and IT-trained users.

In addition, this paper presented two indicators to measure LCDP performance within the software development process: Low Code Factor (*LCF*), which measures the software development effort needed for the app creation using an LCDP, and the *LCDPfit*, a metric that can assess the suitability of an LCDP to realize the intended use cases. These metrics and results can be used by managers and practitioners to support an effective and successful LCDP implementation. The applied research method can be expanded by HR and ISM researchers to support their conceptual artifacts in a low-code development context with data. Also, the suggested indicators can be used to assess the process performance of the software development with LCDP.

In our future work, the focus will be on understanding the intensity of the programming activity and how it might reflect a behavioral pattern. This will involve quantifying the motivation of the developer team by using the activity/action profiles of the app development process. Additionally, we envision exploring, how LCDP empowers BUDs within their working environment. Another future research direction will focus on the job-crafting effects of LCDP-based development for BUD and experts.

REFERENCES

[1] K. Frosch and O. Levina, „Taking the Matter in their own Hands – Can Business Unit Developers Fullfill their Digital Demands with Low-Code Development Platforms?," Design and application of socially-aware IT (DASAIT) IARIA, Apr. 2023, no. 18001, ISSN: 2308-3956, ISBN: 978-1-68558-077-3.

[2] S. Rafi, M. A. Akbar, M. Sánchez-Gordón, and R. Colomo-Palacios, „DevOps Practitioners' Perceptions of the Low-code Trend," International Symposium on Empirical Software Engineering and Measurement, pp. 301–306, Sep. 2022, doi: 10.1145/3544902.3546635.

[3] M. M. Li, C. Peters, M. Poser, K. Eilers, and E. Elshan, „ICT-enabled job crafting: How Business Unit Developers use Low-code Development Platforms to craft jobs," International Conference on Information Systems (ICIS), Dec. 2022, ISBN 978-1-958200-04-9.

[4] K. Talesra, „Low-Code Platform for Application Development," International Journal of Applied Engineering Research, vol. 16, pp. 346–351, doi: 10.37622/IJAER/16.5.2021.346-351, 2021.

[5] M. Hirzel, „Low-code programming models," Communications of the ACM, vol. 66, pp. 76–85, 2023.

[6] A. Trigo, J. Varajao, and M. Almeida, „Low-Code Versus Code-Based Software Development: Which Wins the Productivity Game?," IT Professional, vol. 24, pp. 61–68, 2022, doi: 10.1109/MITP.2022.3189880.

[7] J. Smith, „The Estimation of Effort Based on Use Cases," Rational Software, White Paper. [Online]. Available from: https://www.inf.ufpr.br/andrey/ci221/docs/finalTP171.pdf (last accessed: Dec 1, 2023).

[8] R. Sanchis, Ó. García-Perales, F. Fraile, and R. Poler, „Low-code as enabler of digital transformation in manufacturing industry," Applied Sciences, vol. 10, 12, Dec. 2020, doi: 10.3390/app10010012.

[9] F. Nowak, J. Krzywy, and W. Statkiewicz, „Study on the Impact of the Use of No-code Application on Internal Logistics Processes in a Company from the E-Commerce Industry - Process Analysis," European Research Studies Journal, vol. 25, pp. 59–71, Aug. 2022, doi: 10.35808/ERSJ/2936.

[10] L. Bies, M. Weber, T. Greff, and D. Werth, „A Mixed-Methods Study of Low-Code Development Platforms: Drivers of Digital Innovation in SMEs," International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME), 2022, doi: 10.1109/ICECCME55909.2022.9987920.

[11] T. C. Lethbridge, „Low-Code Is Often High-Code, So We Must Design Low-Code Platforms to Enable Proper Software Engineering," Lecture Notes in Computer Science, vol. 13036, pp. 202–212, 2021, doi: 10.1007/978-3-030-89159-6_14/COVER.

[12] J. Hintsch, D. Staegemann, M. Volk, and K. Turowski, „Low-code Development Platform Usage: Towards Bringing Citizen Development and Enterprise IT into Harmony," ACIS 2021 Proceedings, vol. 10, Jan. 2021. [Online]. Available from: https://aisel.aisnet.org/acis2021/11.

[13] A. Kermanchi, „Developer Experience in Low-Code Versus Traditional Development Platforms - A Comparative Experiment," Aalto University, Dec. 2022 [Online]. Available from: https://aaltodoc.aalto.fi/server/api/core/bitstreams/18f9453c-5930-4a94-a545-4f9dc51be7aa/content (last accessed: Dec 1, 2023).

[14] R. Bernsteiner, S. Schlögl, C. Ploder, T. Dilger, and F. Brecher, „Citizen vs. Professional developers: Differences and Similarities of Skills and Training Requirements for Low Code Development Platform", in ICERI2022 Proceedings, 2022, pp. 4257–4264.

[15] A. Sahay, A. Indamutsa, D. Di Ruscio, and A. Pierantonio, „Supporting the understanding and comparison of low-code development platforms," Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 2020, pp. 171–178.

[16] D. Krejci, L. Küng, and S. Missonier, „A Case Study of Enterprise-wide Digital Innovation: Involving Non-IT Employees", ECIS 2022 Research Papers, June 2022. [Online]. Available from: https://aisel.aisnet.org/ecis2022_rp/55 (last accessed: Dec 1, 2023).

[17] M. Lebens, R. J. Finnegan, S. C. Sorsen, and J. Shah, „Rise of the Citizen Developer," Muma Business Review, Vol. 5, pp. 101–111, 2021, doi: 10.28945/4885.

[18] A. C. Bock and U. Frank, „Low-Code Platform," Business and Information Systems Engineering, vol. 63, pp. 733–740, Dec. 2021, doi: 10.1007/S12599-021-00726-8/FIGURES/1.

[19] E. W. Boot, J. J. G. Van Merriënboer, and A. L. Veerman, „Novice and experienced instructional software developers: Effects on materials created with instructional software templates," Educational Technology Research and Development, vol. 55, pp. 647–666, 2007, doi: 10.1007/s11423-006-9002-9.

[20] C. Bunse, I. Grützner, C. Peper, S. Steinbach-Nordmann, and G. Vollmers, „Coaching professional software developers an experience report," Software Engineering Education Conference, pp. 123–130, 2006, doi: 10.1109/CSEET.2006.11.

[21] H. I. Akyüz and M. Kurt, „Effect of teacher's coaching in online discussion forums on students' perceived self-efficacy for the educational software development," Procedia - Social and Behavioral Sciences, vol. 9, pp. 633–637, Jan. 2010, doi: 10.1016/J.SBSPRO.2010.12.209.

[22] E. Elshan, E. Dickhaut, and P. Ebel, „An Investigation of Why Low Code Platforms Provide Answers and New Challenges," 56th Hawaii International Conference on System Sciences, 2023, ISBN: 978-0-9981331-6-4.

[23] R. Pressman, *Software Quality Engineering: A Practitioner's Approach*, New York, NY, USA: McGraw-Hill Education, 2009.

[24] C. J. Lokan, „Function points," Adv. Comput, vol. 5, pp. 297–347, 2005.

[25] M. K. Chemuturi, *Software Estimation Best Practices, Tools & Techniques*. Fort Lauderdale, J. Ross Publishing, 2009.

[26] R. K. Clemmons, „Project estimation with use case points," J. Defense Softw. Eng., vol. 19, pp. 18–22, 2006.

[27] K. Petersen, „Measuring and predicting software productivity: A systematic map and review," Inf. Softw. Technol., vol. 53, pp. 317–343, 2011.

[28] R. Premraj, B. Kitchenham, M. Shepperd, and P. Forselius, „An empirical analysis of software productivity over time," 11th IEEE International Software Metrics Symposium (METRICS), Sept. 2005, https://doi.org/10.1109/METRICS.2005.8.

[29] Joget. [Online]. Available from: www.joget.org (last accessed: Dec 1, 2023).

[30] M. Ochodek, J. Nawrocki, and K. Kwarciak, „Simplifying effort estimation based on Use Case Points," Information and Software Technology, vol. 53, pp. 200–213, 2011, doi: 10.1016/j.infsof.2010.10.005.

[31] European Commission, Technology readiness levels (TRL), 2014. [Online]. Available from: https://ec.europa.eu/research/participants/data/ref/h2020/

wp/2014_2015/annexes/h2020-wp1415-annex-g-trl_en.pdf (last accessed: Dec 1, 2023).

[32] O. Levina, UCPA for Low Code Factor Calculation. [Online]. Available from: https://zenodo.org/records/8308333, 2023 (last accessed: Dec 1, 2023).