Disentangling the Potentials of Low Code Development Platforms - A Functional Affordance Perspective

Ernestine Dickhaut, University of Kassel, Ernestine.dickhaut@uni-kassel.de
Edona Elshan, University of St.Gallen, Edona.elshan@unisg.ch
Andreas Janson, University of St.Gallen, andreas.janson@unisg.ch

Introduction

In today's world, companies must undergo digital transformation to remain competitive and hence survive in the market (Bexiga et al., 2020). To push the digital transformation, we observe, that many companies are developing digital products. However, the corresponding software development is fraught with difficulties. Cost overruns, conflicting project requirements, overly long development times, or even a lack of business-IT alignment are examples of these (Charette, 2005). Depending on the size of the company, development costs can reach seven-figure sums and thus significantly determine the company's success. Standish Group (2015) outlines that 19% of the analyzed IT projects fail early on, while 52% cause higher costs, require a longer development time, or are associated with other problems. Thus, new techniques and methods such as agile system development and working in SCRUM teams are becoming popular to develop systems as quickly and efficiently as possible, especially with business development goals in mind. In particular, low-code development platforms (LCDP) may be a manner to overcome these hurdles. Unnecessary revisions, which cost a lot of time and money, can be avoided by having the necessary expertise at the beginning of the development process. Programming environments that allow non-coders, such as business developers with domain know-how, to program simple systems may empower non-developers to write small programs on their own. The systems can quickly be adapted with little effort, which is beneficial for agile system development. Within companies, these LCDPs are part of a larger trend of technology democratization (Brinker, 2018), referring to any undertaking that traditionally required coding but can now be accomplished by a business user. Business developers respectively end-user developers (whether in education, marketing, accounting, design, or operations) thus have greater control over their tasks and workflows.

In particular the business users - which we refer to in the paper as citizen developers (CD) - shall profit from the implementation of LCPDs. In accordance with Khorram et al. (2020), the benefit of LCPDs is large that they may be utilized by those who do not have programming experience, such as product managers. Consequently, no product manager ends up in a back-and-forth with developers, and faster higher-quality outputs are promised. Thereby, by utilizing drag-and-drop and light coding skills, the software can be built more quickly and in addition to that, it can be operated more easily from an end-user perspective (Clark, 2019). But LCDPs do not bear only potential for CD but also for software developers. As a result, less effort is required by the IT department when designing software employing LCPDs, resulting in cheaper expenses. Furthermore, there are various other advantages to using LCPDs for CD (Sanchis et al., 2019).

Up to now, no prior research has been undertaken on the extent to which are the affordances of citizen developers and software developers in the development of software utilizing LCDPs. Within our study, we examine the affordances of low code development platforms. By this, we would like to disentangle the affordances by investigating citizen developers (CD) and software developers (SD) as defined above. This disentanglement has two reasons. First, given the high growth rates, LCDP may become a reality for many organizations within the next few years. Second, given the importance of digital products and software being at the heart of products and services (Yoo et al., 2012), developing digital products has become a key organizational activity. Thus, we formulate two research questions (RQ):

> **RQ1:** What affordances do LCDP offer to software developers and citizen developers?
>
> **RQ2:** What are future research directions to investigate LCDPs?

Specifically, we pursue to generate broad insights and centralize the available information. Based on our initial conceptualization of low code and according to LCDPs, we provide a scoping review of extant literature to gain a more substantiated understanding of the affordances of LCDP. We address these research questions through a systematic literature review. Next, we briefly review low code development platforms and the concept of functional affordances. We then present the study and discuss the implications for research and practice.

Research Approach

This study employs a systematic literature review to analyze the research on LCDPs to disentangle the affordances of software developers and citizen developers. This approach is in line with the aim of our study to comprehensively review and summarize the extant literature. In this regard, we followed the steps proposed by Cooper (1988), Fettke (2006), and vom Brocke et al. (2015). We specify the characteristics of our review and classify the SLA according to Cooper (1988) at the start of our SLA. Accordingly, the purpose of our SLA is the identification of central affordances in the state of the art and the generalization of current solutions to the common denominators, which falls under the integration goal category. As a result, the theories, methods, and applications of the solutions, as well as their outcomes, are the emphasis of our SLA. Our coverage is extensive since our goal is to present a generalizable overview of the affordances of low code development platforms for both software developers and citizen developers.

We prepare the database search process as the second stage in our SLA. Thereby, we systematically searched scientific databases to identify articles that address low code development. One characteristic of our systematic literature search is that we use a structured approach in which research findings are identified, evaluated, and subsequently synthesized for presentation (Vom Brocke et al., 2015; Webster & Watson, 2002). The literature search was conducted within the fields of information systems (IS) and computer science (CS) focusing on articles from conferences and peer-reviewed journals to cover a wide range of publications. The keywords and, as a result, the search string is presented below: *"low-code" OR "low code" OR "no-code" OR "no code"*.

We used a qualitative content analysis (Bygstad et al., 2022) to identify the affordances across the research articles. The initial stage was to code data using an open coding technique, focusing on events, important entities, and LCDPs features. We then went through another iteration of coding in which we tied results to essential entities and aspects of LCDPs, to establish relationships between them. We discovered affordances in an iterative procedure, with the three authors working independently to identify affordances, discussing and refining them, then returning to the data to check the modified set of affordances. This procedure was repeated until no further affordances resulted from the iterative approach. While individual affordances have been studied, Strong et al. (2014) and Volkoff and Strong (2013) distinguish basic and higher-level affordances to identify organizational affordances, i.e., prospective business efforts attaining organizational-level instant concrete outcomes in alignment with organizational objectives. As a result, organizational and higher-level affordances vary from the result or impact of affordances: whereas organizational affordances stay consistent across organizations, how and why they are actualized varies among individuals. Therefore, there are numerous ways in which organizational affordances might be realized, resulting in fundamentally varied structures (Strong et al., 2014).

Findings

In the following, the results from the literature analysis are presented. We differentiate technical affordances, design affordances and business affordances following the work by Ostern et al. (2020). In the derivation, we distinguish according to the respective user groups: CDs as laymen with little to no programming experience and SDs as experienced programmers.

Technical affordances are derived from the design features of a technology as well as the purpose and aims of individuals who use it (Leonardi & Treem, 2012). Hence, we describe technical affordances of LCDPs as the association between the features of LCDPs and an organization that determines which behaviors may be enabled by the implementation of the technology, given the organization's intention, abilities, and aims. We discovered three technical affordances when researching LCDPs: exploiting cloud-native potentials, enhancing development process, enforcing system development variety.

A *design affordance* stems from the LCDP use by a user. They do not depend on the technical prerequisites and arise much more during operation. Users are usually not aware of the design affordances in advance and only notice them as use progresses, for example, reduction of development effort. The SD, who is actually used to a lot of coding and complex programming, only realizes over time how LCDP can reduce development effort.

We identified five design affordances across our analysis: increase of productivity, reduction of development effort, increases of software quality, acceleration of activities, user interface design, and development of complex technologies and frameworks that can be adapted in a simplified way.

Low code is frequently used by companies to achieve a specific purpose, mainly internally, such as shortening their own development time. Fields of application where the purpose is external or embedded in an innovative context were rarely or only slightly addressed. We define *business affordances* as the affordances of the whole company. Thus, combining technical and business affordances value can be created through the use of LCDPs. We identified four business affordances: reduction of development costs, acceleration of activities, ad-hoc prototyping and reskilling/ upskilling.

Discussion

In our paper, we conducted a systematic literature review to analyze the affordances of LCDPs. We surveyed scientific and practical-oriented papers, evaluated and synthesized them for the presentation of the results. The main results yield a wide variety of application fields and scenarios where LCPDs plays a significant role. Likewise, we identified eleven affordances related to the use of low code. Our literature review shows that low code can be applied in a wide variety of application domains. From chatbot development to process modeling, but also application scenarios in the financial sector.

As part of our paper, we discuss the contributions of our review and provide an agenda for future research directions. We see it as important to further advance the understanding and deployment of LCDPs, as well as their affordances that we derived in the context of this study. Furthermore, future research should explore seemingly contradictory findings. The relatively small number of publications on low code demonstrates that this is still an underestimated research area although low code technologies are becoming more and more important in various contexts. As described in the introduction, low code is not a novelty, and the basic idea has been around for many years. However, the underlying conditions have changed in a platform-driven software world. We divide our agenda for future research directions into three parts following our functional affordances classification – technical affordances, design affordances and business affordances.

References

Bexiga, M., Garbatov, S., & Seco, J. C. (2020). Closing the gap between designers and developers in a low code ecosystem. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*. ACM. https://doi.org/10.1145/3417990.3420195

Brinker, S. (2018). *Democratizing martech: distributing power from IT to marketing technologists to everyone*.

Bygstad, B., Øvrelid, E., Ludvigsen, S., & Dæhlen, M. (2022). From dual digitalization to digital learning space: Exploring the digital transformation of higher education. *Computers & Education*, 104463. https://doi.org/10.1016/j.compedu.2022.104463

Charette, R. N. (2005). Why software fails [software failure]. *IEEE Spectrum*, *42*(9), 42–49.

Clark, L. (2019). Low-Code Maturity Boosts Efficiency and Helps User Acceptance. *Computer Weekly*.

Cooper, H. M. (1988). Organizing knowledge syntheses: A taxonomy of literature reviews. *Knowledge in Society*, *1*(1), 104–126. https://doi.org/10.1007/BF03177550

Fettke, P. (2006). State of the Art of the State of the Art-A study of the research method" review" in the information systems discipline. *Wirtschaftsinformatik*. https://scholar.google.com/citations?user=zbcgxrnuir4c&hl=en&oi=sra

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*.

Leonardi, P. M., & Treem, J. W. (2012). Knowledge management technology as a stage for strategic self-presentation: Implications for knowledge sharing in organizations. *Information and Organization*, *22*(1), 37–59. https://doi.org/10.1016/j.infoandorg.2011.10.003

Ostern, N., Rosemann, M., & Moormann, J. (2020). Determining the Idiosyncrasy of Blockchain: An Affordances Perspective. *ICIS*(978-1-7336325-5-3). https://eprints.qut.edu.au/205041/

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. https://doi.org/10.3390/app10010012

Standish Group (2015). CHAOS-Report 2015.

Strong, D., Volkoff, O., Johnson, S., Pelletier, L., Tulu, B., Bar-On, I., Trudel, J., & Garber, L. (2014). A Theory of Organization-EHR Affordance Actualization. *1536-9323*, *15*(2), 53–85. https://doi.org/10.17705/1jais.00353

Volkoff, O., & Strong, D. M. (2013). Critical Realism and Affordances: Theorizing IT-Associated Organizational Change Processes. *MIS Quarterly*, *37*(3), 819–834. https://doi.org/10.25300/misq/2013/37.3.07

Vom Brocke, J., Simons, A., Riemer, K., Niehaves, B., Plattfaut, R., & Cleven, A. (2015). Standing on the Shoulders of Giants: Challenges and Recommendations of Literature Search in Information Systems Research. *Communications of the Association for Information Systems*, *37*(8), o. S.

Webster, J., & Watson, R. T. (2002). Analyzing the past to prepare for the future: Writing a literature review. *MIS Quarterly*. https://www.jstor.org/stable/4132319

Yoo, Y., Boland, R. J., Lyytinen, K., & Majchrzak, A. (2012). Organizing for Innovation in the Digitized World. *Organization Science*, *23*(5), 1398–1408. https://doi.org/10.1287/orsc.1120.0771