Low-code Development Platform Usage: Towards Bringing Citizen Development and Enterprise IT into Harmony

Johannes Hintsch  
*Otto von Guericke University Magdeburg, johannes.hintsch@ovgu.de*

Daniel Staegemann  
*Otto von Guericke University Magdeburg, daniel.staegemann@ovgu.de*

Matthias Volk  
*Otto von Guericke University Magdeburg, matthias.volk@ovgu.de*

Klaus Turowski  
*Otto von Guericke University Magdeburg, klaus.turowski@ovgu.de*

Follow this and additional works at: [https://aisel.aisnet.org/acis2021](https://aisel.aisnet.org/acis2021)

**Recommended Citation**  
Hintsch, Johannes; Staegemann, Daniel; Volk, Matthias; and Turowski, Klaus, "Low-code Development Platform Usage: Towards Bringing Citizen Development and Enterprise IT into Harmony" (2021). *ACIS 2021 Proceedings*. 11.  
[https://aisel.aisnet.org/acis2021/11](https://aisel.aisnet.org/acis2021/11)

This material is brought to you by the Australasian (ACIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in ACIS 2021 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
Low-code development platform usage: Towards bringing citizen development and enterprise IT into harmony

Research-in-progress

Johannes Hintsch
Otto von Guericke University Magdeburg
Faculty of Computer Science
Email: johannes.hintsch@ovgu.de

Daniel Staegemann
Otto von Guericke University Magdeburg
Faculty of Computer Science
Email: daniel.staegemann@ovgu.de

Matthias Volk
Otto von Guericke University Magdeburg
Faculty of Computer Science
Email: matthias.volk@ovgu.de

Klaus Turowski
Otto von Guericke University Magdeburg
Faculty of Computer Science
Email: klaus.turowski@ovgu.de

Abstract

The ongoing digitization of our world leads to many areas of our lives being more pleasant and improved. New technologies and paradigms are emerging to support the development of software and systems. Their proliferation not only leads to higher complexity of potential solutions, but also to the problem of finding qualified people. Especially enterprises, which are constantly confronted with this problem, are increasingly considering low-code development platforms (LCDP) to allow the development of software by inexperienced and untrained citizen developers. However, at this point, non-functional requirements, such as performance and security, can require a thorough system understanding. In this work, we identify issues that may occur when citizen developers use LCDPs, allowing to deduce success factors for their implementation. Eventually, this shall help decision makers when introducing LCDPs into their environments.

Keywords Low-code, LCDP, citizen developer, non-functional requirements, success factors.
1 Introduction

All kinds of organizations and even individuals rely on software to drive their digitalization initiatives. Commercial-off-the-shelf (COTS) software exists for application areas with general applicability (e.g., word processing, accounting) as well as specific applicability (e.g., IT service management). However, with increasing automation and efficiency requirements custom software development is required: In order to customize COTS software or develop new application software for areas for which adequate COTS products are not available (Woo 2020).

Professional software engineering requires substantial training. Skilled developers that provide thorough experience in this domain are hard to find (Bexiga et al. 2020). At the same time, enterprises require agility and speed in their application development (Sanchis et al. 2020) in order to automate their business and stay competitive. Meeting business requirements with limited software engineering resources is a challenge.

Vendors have recognized this challenge and introduced so called low-code development platforms (LCDP) (Richardson and Rymer 2014). LCDPs come with low entry barriers for users wanting to develop software because low-level technical details do not need to be considered (Silva et al. 2020). These platforms can be characterized as follows (Bager 2021; Richardson and Rymer 2014):

- Graphical tools are provided to design user interfaces, business logic, and data models.
- Users can employ instantly available components (e.g., user interface widgets or connector components to external web services).
- The full application software lifecycle is supported.
- LCDPs are provided via the cloud.

On the one hand, LCDPs can be of benefit to trained software engineers who are relieved of repetitive tasks and gain efficiency by employing the platforms (Silva et al. 2020). On the other hand, untrained persons can use LCDPs to develop application programs. These so-called citizen developers are enabled to bring automation to their business problems without the help of the scarce resource of software engineers (Vincent et al. 2020).

In this paper, we focus on citizen developers in enterprises. Citizen developers are part of business departments. Thus, as business experts, they have an intricate understanding of their enterprise’s problems. Usually, the backlog of application development projects is long (Rymer and Koplowitz 2019). By enabling citizen developers to create automation solutions for their business problems without or with only limited assistance by enterprise IT, more flexibility and speed can be achieved in much-needed application development (Rymer and Koplowitz 2019; Woo 2020).

In their drive to attain greater market shares, vendors are aggressively targeting business departments directly in order to sell their LCDP products. This may result in multiple LCDPs in one enterprise causing technical debt as well as a source of shadow IT (Richardson and Rymer 2016; Vincent et al. 2021). And although citizen developers are assisted by platforms to produce quality software, a substantial qualification gap compared to trained software engineers exists (Barricelli et al. 2019; Khorram et al. 2020; Silva et al. 2020). Because of these factors, representatives of the enterprise IT department, stewards of the enterprise’s information system’s stability, may be skeptical of the autonomous low-code endeavors of their business colleagues. Since this is a problem that cannot be solved technically, we pose the following overarching research question:

*What are organizational conditions under which citizen developers can leverage LCDPs to the business’ benefit without neglecting non-functional IT requirements?*

The focus is on non-functional requirements because functional requirements are addressed by the LCDP vendors.
In the next section, the research background is provided. The research process is described in section three. The contribution of this paper, issues of citizen development from the IT department’s perspective, is presented in section four. The final section makes concluding remarks to this research-in-progress paper.

2 Background

Attempting to simplify software development through higher levels of abstraction is not new. Computer-assisted systems engineering tools (CASE) appeared in the 1980s and 1990s (Mew and Field 2018). Similarly to 4th generation programming languages (4GL), these tools never had large commercial success (Richardson and Rymer 2016). This is in part due to their proprietary approaches that restricted their interoperability to their vendors’ ecosystems, e.g., because of non-open application programming interfaces. This so far missing openness is a benefit of current LCDPs (Richardson and Rymer 2016). In addition, the modern platforms are easy to understand for untrained users (Mew and Field 2018). However, their abstraction introduces an additional system layer that increases the overall complexity and, thus, may lead to malfunction (Bager 2021).

The idea to enable users without a formal software engineering education to perform programming is also researched under the terms of: end user development, end user programming as well as end user software engineering (Barricelli et al. 2019). On the one hand, end user development has a rather simplistic approach and can be found in such technologies such as end user-friendly scripting for home automation. On the other hand, end user software engineering tries to bring the full rigor of the software engineering discipline to end users. These approaches are not targeted at cloud platforms, but may be found in all sorts of tool deployments with varying techniques, such as rule-based, spreadsheet-based, natural language, or gesture-based (Barricelli et al. 2019).

Silva et al. (2020) studied how citizen developers (the synonymous term for end user in the context of LCDPs) perform using LCDPs in contrast to trained software engineers. The platforms must make a trade-off between the ease of use for untrained citizen developers and being useful to trained software engineers. They found out that the experts had trouble in identifying key software engineering concepts in LCDPs, which may result in inefficient use of LCDPs. Citizen developers faced issues with less intuitive tasks such as creating database connections, program parameter passing, as well as screen creation. LCDPs also support collaborative work by citizen developers and trained software engineers (Richardson and Rymer 2016). In such cases, trained software engineers may create custom components for a specific use case that can then be integrated by citizen developers into their software projects.

Various successful use cases are reported in scientific and industry literature. Woo (2020) describes the case of rapid development of an application of an LCDP for COVID-19 related operations. Dushnitsky and Stroube (2021) discuss low-coding by e-commerce entrepreneurs who used the LCDP Shopify and became often more successful than competitors with full-fledged in-house software development operations. Various internet of things-related research projects (Korkan et al. 2020; Tisi et al. 2019) report the successful application of LCDPs. The success of LCDPs reported in scientific literature is paralleled by industry analysts. Gartner Inc. projects 50% of all medium and large enterprises will use LCDPs as one of their strategic software platforms by 2023 (Vincent et al. 2020). In their magic quadrant, they list Appian, Mendix, Microsoft, OutSystems, Salesforce, and ServiceNow as leaders. While Appian, Mendix, and OutSystems are pure LCDP vendors, Microsoft, Salesforce, and ServiceNow are not. Microsoft offers integrations with various other parts of its portfolio. Salesforce and ServiceNow are both software as a service providers for their COTS product and provide LCDP not only as a standalone platform as a service offering, but primarily to customize and extend their standard software offering (Vincent et al. 2020). Although Richardson and Rymer (2016) underline the openness of current LCDPs, the interoperability of the platforms, for example in terms of process model standardization, is limited (Ihirwe et al. 2020).
Similar to LCDPs, model-driven engineering tries to enhance development efficiency (Tisi et al. 2019). Bexiga et al. (2020) show how to map and transform design artefacts created by user interface designers into low-code user interface components. They state that this model transformation can save between 20-75% of time invested in style guides. Similarly, but not directed at user interface design, the Lowcomote project attempts to combine model-driven engineering concept with LCDPs (Tisi et al. 2019).

Zolotas et al. (2018) and Nunes Alonso et al. (2020) both address the generation of interfaces to connect systems in the LCDP context. Zolotas et al. (2018) focus on the aspect of security. They criticize the prevailing use of basic authentication means by LCDP coupled with very limited access control mechanisms. Consequently, they propose a REST web service generation mechanism with an elaborate authentication and access control scheme. NoSQL data stores are widely used in today’s web application software. However, they do not possess standard query languages. Therefore, Nunes Alonso et al. (2020) propose an interface mechanism for connecting NoSQL data stores for LCDPs.

Ihirwe et al. (2020) and Woo (2020) point to the limited testing and debugging capabilities of LCDPs. Jacinto et al. (2020) and Khorram et al. (2020) focus on testing for LCDPs. Already in traditional development projects, testing is often neglected. Providing LCDPs to novice programmers may increase this issue for LCDP-based application software. Jacinto et al. (2020) show how LCDP-based application software can be unit tested. Because such software usually has multiple dependencies the important unit testing is difficult to implement. Therefore, the authors propose a way of mocking dependent components, a concept also found in traditional development projects (e.g., for database dependencies). Khorram et al. (2020) point to the difficulties of performing testing with citizen developers who have the business expertise to write test cases, but not the technical knowledge to automate them. They propose to assist citizen developers with recommender systems in defining correct automated tests.

The current state of the art focuses on technological concepts of LCDPs. However, no studies could be identified that guide on how to use LCDPs in enterprises (e.g., success factors for implementing LCDPs). Older studies, e.g., on CASE tools, will not feature the needed insights because CASE does not include the concepts similar to the citizen developer.

3 Research approach

In order to answer the research question, we use a segmented approach as depicted in Figure 1. After the Google Scholar-based literature search (keywords: “low code” and “no code”, only papers with citations, no patents), where the state of the art of low-code research was identified, the problem statement was defined. Because LCDP are quite new, academic and grey literature was considered. In order to find an answer to the research question, we first identify the issues of applying the citizen developer concept with LCDPs. In order to do so, a conversation with an industry expert was conducted. The conversation was guided by non-functional IT requirements. This issue identification forms the content of this paper.

![Figure 1: Research process](image-url)
In the full research, a case study (interview-based) will be carried out to study the application of LCDPs with citizen developers with a focus on the issues that occur in regards to the non-functional requirements of IT. The results should serve to validate or invalidate the issues identified. These validated issues are the basis for the solution design of organizational conditions, which must be met in order to enable citizen developers to leverage LCDPs to the business’ benefit without neglecting non-functional IT requirements.

4 Issues of citizen development from the IT department’s perspective

This section starts by outlining the motives of business departments, to which citizen developers belong.

Business departments require the prioritization of their feature and change requests and have limited understanding for the IT department’s processes to prioritize all business departments’ feature and change requests. Agility and good quality as well as an understanding for the underlying business problems, business and process expertise are required (Luftman and Brier 1999). Conversely, non-functional requirements such as reliability, testability, understandability, modifiability, performance, and security (Glinz 2007) must also be considered, despite oftentimes not being the business department’s main focus. With the help of an industry expert (LCDP implementer, three years of LCDP experience & ten+ years of IT experience), issues that may occur when citizen developers use LCDPs where gathered. The non-functional requirements were used as a guiding structure for the discussion process. Table 1 summarizes the identified issues.

Service level agreements are provided by IT departments for their services in order to provide some reliability and predictability to the business. However, if these LCDP-based application software services are no longer fully coming from the IT department, such agreements cannot be made. The IT department can provide service level agreements for the LCDP, but not for the application software which the IT department did not develop.

<table>
<thead>
<tr>
<th>Area</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issue 1 Service level agreements</td>
<td>Service can only be provided for the LCDP, not for the LCDP-based application software</td>
</tr>
<tr>
<td>Issue 2 Integration into support structure</td>
<td>Who will be the contact for issues with the application software</td>
</tr>
<tr>
<td>Issue 3 Testing</td>
<td>Testing must be properly performed in order to ensure stable software, citizen developers are not fully trained to write tests</td>
</tr>
<tr>
<td>Issue 4 Security</td>
<td>Corporate and general security practices may not be as efficiently implemented by citizen developers as by trained software engineers from the IT department</td>
</tr>
<tr>
<td>Issue 5 Modifiability (Documentation)</td>
<td>Documentation is critical in order to understand and maintain software. If this is not properly done maintenance is made difficult</td>
</tr>
<tr>
<td>Issue 6 Performance</td>
<td>Computational complexity is a focus area of software engineering training. If this is not properly regarded performance may be hindered</td>
</tr>
<tr>
<td>Issue 7 Duplication</td>
<td>Enterprise architects guide the development of the enterprise information system. They prevent duplication of functionality and data</td>
</tr>
</tbody>
</table>

*Table 1. Issues of citizen developers in LCDP*
Related to service level agreements are the service levels with a support structure. 1st to 3rd level support organizations usually assist with problems in regards to application services. If the responsible developers are citizen developers from business departments, responsibilities are unclear compared to conventional settings.

As stated by previous works, testing requires software engineering competence. And, especially automated testing is important, for instance, in continuous deployment concepts.

Security practices are provided by security officers and IT departments are experienced in implementing them. However, this missing experience and the direct own benefit of quick implementation without considering possible adverse effects on security may mean that citizen developers will not implement security as effectively as their software engineering trained colleagues.

Documentation is a tedious, but necessary task of software engineers, especially when application software goes into operation. If the documentation is not created it is difficult to understand the underlying software mechanisms. Limited software understandability hinders software maintainability.

The performance of an application can be negatively affected by bad programming. For instance, implementing unnecessary database lookups can be considered bad programming. Software engineers are trained in concepts of computational complexity and should avoid these issues. If such awareness is not present, citizen developers are likely to program software with low performance. Although the LCDPs may themselves detect some of those issues, performance could be decreased when untrained citizen developers program their own application software.

Enterprise architects guide the development of the enterprise’s information system and can prevent duplication of functionality and data storage. If citizen developers autonomously develop application software, it is however highly likely that they do so without considering or even just knowing the bigger picture. In turn, functionality duplication may lead to inconsistent process behaviours and data storage to inconsistent data on an enterprise level.

5 Concluding remarks

We identified seven issues that may occur with applying the citizen developer concept to LCDPs in an enterprise. Such issues are avoided by a rigorous IT department, the stewards of the non-functional IT requirements.

In future work, these issues must be studied and validated. We propose a case study approach for this. For validated issues, organizational conditions must be found, e.g., through process patterns, to prevent these issues from occurring. With these organizational conditions, the overarching research question can be answered and subsequently success factors deduced.

6 References


