Human Systems Engineering and Design (IHSED2023), Vol. 112, 2023, 426–435
https://doi.org/10.54941/ahfe1004155

How to Select and Implement a Suitable Low-Code Development Platform

Sven Hinrichsen, Alexander Nikolenko, Kai Leon Becker, and Benjamin Adrian

Industrial Engineering Lab, Ostwestfalen-Lippe University of Applied Sciences and Arts, Lemgo, Germany

ABSTRACT

Low-code programming allows the creation of software applications using a graphical user interface with minimal classical programming code ("low code") and without requiring extensive programming knowledge. This puts it in contrast to previous generations of programming languages. The advantages of low-code development are manifold, including the increase of software development capacities through a partial decentralization of the development process, speeding up software development through the low-code approach, and designing software with a strong user-centric focus. Using a low-code development platform can help companies adapt their own business processes to changing requirements more quickly and to make complexity resulting, for example, from heterogeneous customer wishes, manageable. Since many low-code development platforms are available, it is not easy for companies to select and successfully introduce a platform that meets their requirements. For this reason, this article presents a procedure model that assists in the process of selecting and implementing a platform.

Keywords: Complexity management, Low-code development platform, Process model for selection and implementation

INTRODUCTION

As markets and customer needs continue to evolve, the emergence of new technologies or changed organizational principles leads time and time again to paradigm shifts in production system design. Especially the trend towards configuration and individualization of products by customers, as well as the integration of ever more functions into products – for example in the sector of mechanical engineering (Brecher et al., 2011) – in conjunction with shorter product life cycles have led to a significant increase in complexity (e.g., Theuer and Lass, 2016; Schuh et al., 2017). This, in turn, led to a significant increase in the amount of information companies need to process. Every customer order in multi-variant batch production, and even more so in industrial individual production (customization), comes with particular requirements. These affect the entire product development and order fulfillment process. Therefore, new approaches are needed to deal with this complexity (Hinrichsen and Bornewasser, 2020). The goal, on the one hand, is to keep complexity costs low (Hvam et al., 2020), on the other, to offer

customers a sophisticated range of products and services (Ponn and Lindemann, 2011). For many decades, the focus has been on strategies of avoidance and reduction of complexity (see Figure 1). On the one hand, these were applied to the range of products and services offered on the market, and on the other, to operating processes (Hvam et al., 2020). However, in variant-rich batch production and especially in individual industrial production, strict limitations are placed on these strategies of complexity avoidance and reduction, as customers often expect that even particular requirements for products and services are taken into account. Therefore, a strategic reorientation should accompany industrially oriented individual production. The goal is to better focus on strategies for mastering complexity. Complexity is seen as inherent to the system (Brinzer and Banerjee, 2017; Hinrichsen and Bornewasser, 2020). Complexity can be mastered through different strategic approaches. These concern personnel, organization and technology, as well as the compatibility between all system elements (see Figure 1). The intended result of this approach of mastering complexity is to improve a company's agility and thus enable it to cope with ever-changing requirements.

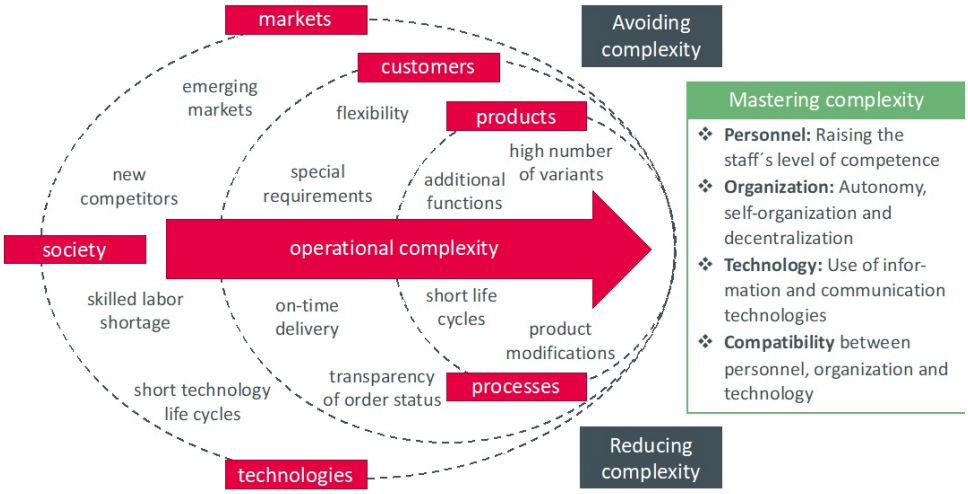

Figure 1: Strategies for managing complexity.

The personnel approach to mastering complexity is concerned with the employees' competencies. Its goal is to enable them to cope with varying and changing requirements and to refine their own work processes. The second, organizational approach to managing complexity originates from systems theory (Kirchhof, 2003). It follows from this theory that highly complex work-sharing systems usually cannot centrally process the large amounts of information which are externally and internally generated from the most varied and changing external requirements. Central organizational units would often lack the flexibility and knowledge to deal with the varying demands of different stakeholders. Therefore, autonomy, self-organization and decentralization would represent an important approach to mastering complexity

(Kirchhof, 2003). A third approach to mastering complexity involves using information and communication technologies. In the context of the Industry 4.0 concept, these are seen as a guarantee for effectively and efficiently processing the large, dynamically developing volumes of information associated with individual or multi-variant batch production, while at the same time keeping unit costs low. The fourth approach to mastering complexity is to accommodate for the compatibility principle. This principle describes the fit between the elements of a socio-technical system and represents the prerequisite for a successful interaction of all elements while working towards a goal (Bläsing et al., 2021).

LOW-CODE SOFTWARE DEVELOPMENT AS AN APPROACH TO MASTERING COMPLEXITY

Low-code programming allows the creation of software applications using a graphical user interface with minimal classical programming code ("low code") and without requiring extensive programming knowledge. The code is generated automatically in the background or is respectively stored in the selected individual function modules (Kahanwal, 2013; Waszkowski, 2019). Programming is done via a low-code development platform (LCDP), which constitutes the development environment and is usually cloud-based (Sanchis et al., 2020; Sahay et al., 2020). The development of application software via an LCDP – sometimes also referred to as end-user development – can decisively contribute to making complexity in companies manageable.

Low-code development implicitly incorporates the four principles for mastering complexity described above (see Figure 1). The use of LCDP makes it possible to at least partially decentralize software development (organizational approach to mastering complexity). Such decentralization has several advantages. For example, it usually takes companies a very long time to develop application software centrally and code-based. This is due to a shortage of IT specialists with longtime experience on the labor market, which often creates bottlenecks in IT departments. In addition, there are often interface and communication problems between the future software users and the programmers from the central IT department. Decentralizing development and thus shifting tasks to specialist departments can therefore help to increase in-house programming capacities and reduce existing communication problems. As a result, the time needed for software development projects can be shortened. By decentralizing software development through low-code, it is also possible to rapidly adjust existing software to changing customer and user requirements.

In order to at least partially decentralize software development by means of an LCDP, IT-savvy employees from specialist departments must be enabled to program applications using such a platform (personnel approach to mastering complexity). This requires a modular training concept to establish qualification standards for low-code programming. To ensure a high quality of software development, a company-specific guideline should be created to outline all software development standards that apply to the selected LCDP.

For example, these standards may include guidelines for designing graphical user interfaces and conducting software tests.

In addition to decentralization and empowerment, a third approach to mastering complexity is to use information and communication technologies to process, store and transmit large, dynamically changing amounts of data. LCDPs address this growing demand for company-specific software applications for processing information and digitizing processes. Furthermore, they support the trend towards the use of apps on mobile devices by creating essential prerequisites for programming "enterprise apps" (Gröger et al., 2013).

In accordance with the compatibility principle, an LCDP that is compatible with both the requirements of the employees and the company (fourth approach to mastering complexity) is to be selected. For this, the complexity-compatibility paradigm (Karwowski, 2005) must be taken into account, which states that complete compatibility can never be achieved, and that incompatibilities can only ever be minimized. In view of this principle, Lethbridge (2021) finds fault with the fact that, despite the term "low code", some LCDPs actually require the generation of large amounts of code for software development, and that the maintenance of this code can be more difficult than with classical programming languages, as LCDPs do not adequately support best practices such as versioning, work-sharing development, reuse of program modules or automated testing. Therefore, it is vitally important for companies to select an LCDP that meets their requirements in order to minimize incompatibilities.

PROBLEM STATEMENT, OBJECTIVE AND METHODS

According to market researchers, there are now more than 200 different LCDPs available. These differ in terms of their functional orientation, application focus, and underlying technology (Al Alamin et al., 2021). Given the very wide and confusing range of LCDPs on the market, it is challenging, especially for SMEs, to select and successfully implement a platform that meets their requirements to a high degree. Currently, there is no method available that assists in the operational selection and implementation process of such a platform. The choice of an LCDP is a strategic decision, as it has a long-term character and can only be reversed with great effort. Once a company has programmed and implemented numerous applications, the hurdle for changing the platform provider is quite high.

This paper aims to present a procedure model for selecting and introducing an LCDP. This model supports companies in finding a platform that meets their requirements and in implementing it successfully in the company. The model consists of six phases, to each of which are assigned recommendations for actions and tools. The procedure model is based on the phases of the REFA standard program on work system design (REFA, 2015). The model was developed and tested within the context of two company projects focused on selecting and implementing of an LCDP.

RESULTS

The process model for selecting and introducing an LCDP consists of six phases (see Figure 2). The individual phases are in turn subdivided into steps. Since the task of selecting and introducing an LCDP is of strategic significance for many companies and also concomitant with a high degree of complexity, that task is organized as a project. In the first phase, therefore, the initial situation is analyzed and the project framework is defined. This framework includes a brief description of the initial situation (step 1.1), a definition of the project goals (step 1.2), a delimitation of the project contents (step 1.3), appointments to the project team (step 1.4), a plan for resource allocation (step 1.5) and a milestone plan (step 1.6). As a result of the first phase of the process model, all participants have a mutual understanding of the project's contents and framework.

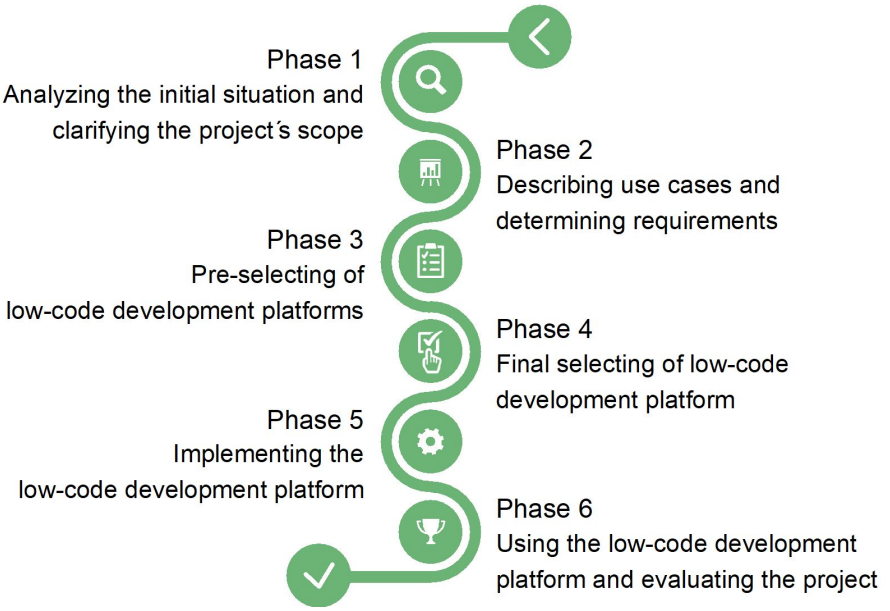

Figure 2: Process model for selecting and introducing a low-code development platform.

The second phase of the procedure model consists of three steps. In step 2.1, interviews or workshops are conducted in particular departments of the company to identify potential use cases. Criteria for possible use cases are, for example, business processes that the interviewees consider to be administratively time-consuming (e.g., forwarding documents by e-mail as part of an approval process), wasteful (e.g., waiting times, duplication of work, errors) or involve media discontinuities (e.g., copying data from the ERP system to a spreadsheet program daily). The identified use cases are systematically described in step 2.2. To that end, the processes which are to be optimized are documented on a macro level. From the outlined use cases, requirements for selecting the LCDP are defined in step 2.3. These requirements are discussed, complemented and finalized in a workshop.

The aim of the third phase of the procedure model is to filter out a few platforms from the large number of LCDPs offered on the market. These few platforms should, in principle, be able to meet the requirements of the company. In this third phase of the procedure model, again three steps are taken. In step 3.1, it is determined which of the many platforms offered on the market should be considered in the analysis process, since it is impracticable for almost every company to concern itself with the characteristics of more than 200 platforms. Bratincevic and Rymer (2020) recommend using a platform offered by a market-leading provider. They justify this advice by stating that market-leading providers usually have many years of experience. An extensive customer portfolio also indicates the reliability of a provider and the attractiveness of its offer (Bratincevic and Rymer, 2020). Accordingly, the degree of market penetration of an LCDP can be used as the first selection criterion. Market analyses with estimates of the market share of individual platform companies are provided by the market research companies Forrester Research (Koplowitz and Rymer, 2019) and Gartner. Forrester Research and Gartner list over 50 vendors when their analyses are combined. This list of providers can be supplemented with platform providers that are not listed by the market research companies but are considered established in the respective national market. A further selection is to be made in step 3.2. With regard to the providers listed by Forrester Research and Gartner, it is recommended only to consider those that are classified as "Leaders" or "Strong Performers" by Forrester Research or as "Leaders" or "Challengers" by Gartner. In addition, further selection criteria for the pre-selection are to be formulated, considering the requirements determined in phase 2 of the model. Step 3.3 involves pre-selecting platforms through methods such as utility analysis, resulting in a shortlist of three to eight platforms.

The fourth phase of the process model aims to make a final selection of a platform based on the preselected platforms. In step 4.1, the criteria for a final selection are defined. Step 4.2 involves obtaining offers and testing the shortlisted platforms. Based on the assessments made and the offers received, a platform is conclusively selected in step 4.3. The criteria which have been used in the operational projects are listed below.

- Open source: Is the source code of the platform openly accessible?
- Application types: What application types of software (mobile, browser, desktop) can be developed?
- Programming language for extensions: Which programming language can be used to extend the LCDP if required?
- Version management: Does the LCDP have its own version management?
- Templates: Does the platform offer the use of ready-made dashboards, diagrams and forms?
- Collaboration options: Can multiple developers work on the same application simultaneously?
- Integration possibilities: Which protocols/ interfaces are supported by the platform?
- Modifiability of the source code: Is it possible to edit the source code?

- Modelling language: Can standardized modelling languages, such as UML or BPMN, be used?
- Deployment options: What deployment options does the platform offer (e.g., on-premises, SaaS, or cloud ERP)?
- Deployment on local infrastructures: Is it possible to deploy the platform software on local infrastructure?
- Location of the cloud servers: In which country are the cloud servers located?
- Location of the company: In which country is the provider located?
- Customer support: How and under what terms is support provided?
- Training courses: What training options are offered? At which terms are training courses offered?
- Experience: Does the company already have experience with the platform provider? Is another software product already being used by the provider?

In the literature, a tabular representation is often used for a detailed comparison of platforms. This can be seen in the works of several authors who compare characteristics of LCDPs and present the availability of features in tabular form (Born, 2019; Ihirwe et al., 2020; Farshidi and Jansen, 2021; Sahay et al., 2020). Such tables can assist in the selection process.

The fifth phase of the model comprises the introduction of the selected LCDP in the company. Throughout the entire introduction process, it is essential to ensure that the platform is well accepted by the employees, by especially emphasizing its benefits. The phase is divided into three steps. In step 5.1, an organizational and qualification concept is developed. The purpose of the organizational concept is to define roles and assign employees to these roles. The following roles are generally to be distinguished: Administrators, experienced developers, citizen developers, and users. Administrators are responsible for the configuration of the platform. They can grant all access rights and create, delete and edit user accounts. The creation of software development guidelines may also be their responsibility. Furthermore, they are in close contact with the provider's third-level support and are responsible for the maintenance of the software. The programs are primarily created by the citizen developers. These work in specialist departments. Particularly suitable are employees who possess the necessary knowledge of their department's processes and are motivated by their affinity for technology to program applications using the platform. Experienced programmers from the IT department are responsible, for example, for developing connectors or supporting citizen developers with complex tasks. Users have read-only rights and can operate the created applications, but not change them. Depending on the assigned roles, different qualification levels should be offered. In addition, qualification should start at the highest role level, i.e., with administrators and experienced developers. In step 5.2, the organizational and qualification concept is implemented. Pilot projects will be initiated to develop first application programs using the platform. Based on the experience gained with these pilot projects, the concept will be implemented in the entire company in step 5.3.

The subject of the sixth phase of the process model is to establish the platform in the company. The further development of the organizational and qualification concept is also a part of this phase. In step 6.1, the existing continuous improvement process is used to ensure that new use cases for low-code programming keep on being identified. In step 6.2, the organizational and qualification concept is to be evaluated and further developed. For example, attended training courses are to be evaluated by the participants. Finally, the object of step 6.3 is to ensure the sustainable success of low-code programming by monitoring even after the end of the project. For this purpose, goals and key metrics can be introduced (e.g., number of low-code apps developed over time; assessment of user satisfaction with individual low-code apps via a standardized questionnaire; estimated cost savings through the introduction of individual low-code apps; number of active citizen developers in the company).

CONCLUSION

Especially the trend towards configuration and individualization of products by customers as well as the integration of more and more functions into products, combined with shorter product life cycles, have led to a considerable increase in complexity. This, in turn, led to a significant increase in the amount of information companies need to process. The success of complexity avoidance and reduction strategies is often limited against the backdrop of heterogeneous customer requirements. Therefore, companies must be able to master high complexity. Low-code development is an approach that can help companies cope with this complexity, meaning large amounts of information needing to be processed. Many business processes can be optimized in a short time by developing suitable low-code applications. Ultimately, information can be efficiently gathered, processed, stored and distributed via such software applications. The advantage of low-code programming lies primarily in its effect on software development, which can be partially decentralized via this approach, so that additional personnel capacities are available for agile and user-centered software development. However, since there is a large number of LCDPs on the market, it is not easy to choose a platform that meets the requirements. The process model presented and tested can help companies select and successfully implement a suitable platform.

ACKNOWLEDGMENT

This publication is a result of the Pro-LowCode project, funded by the Ministry of Economic Affairs, Industry, Climate Action and Energy of the State of North Rhine-Westphalia, Germany (grant no. 005-2011-0021).

REFERENCES

Al Alamin, A., Malakar, S., Uddin, G., Afroz, S., Haider, T., & Iqbal, A. (2021). An Empirical Study of Developer Discussions on Low-Code Software Development Chal-lenges. In 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR), Madrid, Spain, pp. 46–57. https://doi.org/10.1109/MSR52588.2021.00018

Bläsing, D., Bornewasser, M., & Hinrichsen, S. (2021). Cognitive compatibility in modern manual mixed-model assembly systems. Z. Arb. Wiss., pp. 289–302 https://doi.org/10.1007/s41449-021-00296-1

Born, A. (2019). Nieder mit dem Code. iX Magazin, 8(2019), S. 82–87.

Bratincevic, J., & Rymer, J. R. (2020). When And How To Modernize Core Applications Using Low-Code Platforms. Forrester Research. https://www.forrester.com/report/When-And-How-To-Modernize-Core-Appl ications-Using-LowCode-Platforms/RES155943

Brecher, C., Kolster, D., & Herfs, W. (2011). Innovative Benutzerschnittstellen für die Bedienpanels von Werkzeugmaschinen. ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb, 106(7–8), 553–556. https://doi.org/10.3139/104.110607

Brinzer, B., & Banerjee, A. (2017). Komplexitätsbewertung im Kontext Cyberphysischer Systeme. ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb, 112(5), 341–345. https://doi.org/10.3139/104.111709

Farshidi, S., Jansen, S., & Fortuin, S. (2021). Model-driven development platform selection: four industry case studies. Software and Systems Modeling, 20(5), 1525–1551. https://doi.org/10.1007/s10270-020-00855-w

Gröger, C., Silcher, S., Westkämper, E., & Mitschang, B. (2013). Leveraging Apps in Manufacturing. A Framework for App Technology in the Enterprise. Procedia CIRP, 7, 664–669. https://doi.org/10.1016/j.procir.2013.06.050

Hinrichsen, S., & Bornewasser, M. (2020). Veränderung der Gestaltungsparadigmen industrieller Montagearbeit. In M. Bornewasser & S. Hinrichsen (eds), Informatorische Assistenzsysteme in der variantenreichen Montage. Springer Vieweg, S. 1–20. https://doi.org/10.1007/978-3-662-61374-0_1

Hvam, L., Hansen, C. L., Forza, C., Mortensen, N. H., & Haug, A. (2020). The reduction of product and process complexity based on the quantification of product complexity costs. International Journal of Production Research, 58(2), 350–366. https://doi.org/10.1080/00207543.2019.1587188

Ihirwe, F., di Ruscio, D., Mazzini, S., Pierini, P., & Pierantonio, A. (2020). Low-code engineering for internet of things. Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings. https://doi.org/10.1145/3417990.3420208

Kahanwal, B. (2013). Abstraction Level Taxonomy of Programming Language Frame-works. International Journal of Program Languages and Applications, 3(4), 1–12. https://doi.org/10.48550/arXiv.1311.3293

Karwowski, W. (2005). Ergonomics and human factors: the paradigms for science, engineering, design, technology and management of human-compatible systems. Ergonomics, 48(5), 436–463. https://doi.org/10.1080/00140130400029167

Kirchhof, R. (2003). Ganzheitliches Komplexitätsmanagement – Grundlagen und Methodik des Umgangs mit Komplexität im Unternehmen. Springer Fachmedien. https://doi.org/10.1007/978-3-663-10129-1_4

Koplowitz, R., & Rymer, J. R. (2019). The Forrester WaveTM: Low-Code Development Platforms For AD&D Professionals. Forrester Research. https://www.forrester.com/report/The-Forrester-Wave-LowCode-Developm ent-Platforms-For-ADD-Professionals-Q1-2019/RES144387

Lethbridge, T. C. (2021). Low-Code Is Often High-Code, So We Must Design Low-Code Platforms to Enable Proper Software Engineering. In T. Margaria & B. Steffen (eds), Leveraging Applications of Formal Methods, Verification and Validation. ISoLA 2021. Lecture Notes in Computer Science, vol. 13036. Springer, Cham. https://doi.org/10.1007/978-3-030-89159-6_14

Ponn, J., & Lindemann, U. (2011). Konzeptentwicklung und Gestaltung technischer Produkte – Systematisch von Anforderungen zu Konzepten und Gestaltlösungen. 2. Aufl. Berlin, Heidelberg: Springer.

REFA-Bundesverband e. V. (2015). Industrial Engineering: Standardmethoden zur Produktivitätssteigerung und Prozessoptimierung. 2. Aufl. München: Hanser.

Sanchis, R., García-Perales, Ó., Fraile, F., & Poler, R. (2020). Low-Code as Enabler of Digital Transformation in Manufacturing Industry. Applied Sciences. 10(12). https://doi.org/10.3390/app10010012

Sahay, A., Indamutsa, A., Di Ruscio, D., & Pierantonio, A. (2020). Supporting the understanding and comparison of low-code development platforms. In IEEE 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp. 171–178. https://doi.org/10.1109/SEAA51224.2020.00036

Schuh, G., Rudolf, S., Riesener, M., Dölle, C., & Schloesser, S. (2017). Product production complexity research: Developments and opportunities. Procedia CIRP, 60, 344–349. https://doi.org/10.1016/j.procir.2017.01.006

Theuer, H., & Lass, S. (2016). Mastering complexity with autonomous production processes. Procedia CIRP, 52, 41–45. https://doi.org/10.1016/j.procir.2016.07.058

Waszkowski, R. (2019). Low-code platform for automating business processes in manufacturing. IFAC PapersOnLine, 52(10), 376–381. https://doi.org/10.1016/j.ifacol.2019.10.060