The Rise of Generative AI in Low Code Development Platforms – An Analysis and Future Directions

Olivia Bruhin
University of St.Gallen
olivia.bruhin@unisg.ch

Ernestine Dickhaut
University of Kassel
ernestine.dickhaut@uni-kassel.de

Edona Elshan
University of St.Gallen
edona.elshan@unisg.ch

Mahei Manhai Li
University of Kassel
mahei.li@uni-kassel.de

Abstract

This study investigates the relationship between Generative AI (GenAI) and Low Code Development Platforms (LCDPs), providing preliminary insights into Gen's transformative potential in this context. It is based on expert interviews and provides insight into the changing landscape of LCDPs influenced by GenAI. The findings highlight the promising benefits of GenAI in LCDPs, such as increased efficiency and decreased errors, while also emphasizing the importance of human oversight and collaboration. The findings also highlight the importance of interpersonal skills in IT, even in an increasingly automated environment. While the economic efficiency and broader implications of GenAI are still being investigated, the study lays the groundwork for future research in this rapidly evolving domain.

Keywords: Generative AI; Low Code Development, Research Directions

1. Introduction

"Humans are tool users; to deny this is to deny our existence. We are now presented with a new set of tools, and a de facto set of code-production techniques […]"
(Kendon et al., 2023)

In the contemporary digital epoch, the ubiquity of Artificial Intelligence (AI) has expanded beyond the realm of computer science, captivating societal curiosity. An increasing number of individuals are delving into AI technologies AI-resources such as ChatGPT, DALL-E, and Codex. System development, contributing to a significant transformation of system development over the years. From new methodological approaches such as agile, scrum or DevOps to low code or no code and developing code with little or no independently written programming code. Now - in the era of generative AI, new opportunities for system development change are emerging. AI can be used in many different areas. (Kendon et al., 2023) describe that as the following *"AI code generators, along with the plethora of available code on the Internet and sites that facilitate contract cheating, are a striking contrast to the heroic notion of programmers toiling away to create artisanal code from whole cloth."*.

In practice, the potential of generative AI for low code development platforms (LCDP) has been discovered in various ways, and new generative AI features are increasingly announced coming during the next months. LCDPs are development environments for software that enable non-coders to develop high-quality software in a short time (Sanchis et al., 2019).

LCDPs build on existing development approaches streams, such as model-driven engineering (MDE), and combines previous concepts, such as rapid application development and computer-aided software-engineering (Di Sipio et al., 2020). These engineering approaches share several core principles. Bucaioni et al. (2022) mention the core development

URI: https://hdl.handle.net/10125/107321
978-0-9981331-7-1

principles of abstraction, automation, visual notations, and agility.

LCDPs enable rapid and agile development of new IT artifacts and require low technical understanding, which is often prevalent in business (Pantelimon et al., 2019). They support the development of user interfaces, business logic, and data services, and improve productivity at the expense of portability across vendors, as compared with conventional application platforms (Richardson & Rymer, 2016). LCDPs are usually cloud-based services (often categorized as platform-as-a-service) which enable the development of applications with the aid of specially designed tools. LCDPs thereby use prefabricated components and settings configurations to minimize the need for manual programming (Khorram et al., 2020).

The aim of this research is to understand how LCDPs are influenced by recent advancements of Artificial Intelligence Generated Content (AIGC). As a result of this study, we highlight current advancements in LCDPs by describing key opportunities and challenges that could occur when integrating AIGC in LCDPs. additionally, we outline future developments of making LCDPs more effective and efficient using AIGC. Thus, results the following research questions:

RQ 1: What opportunities and challenges of generative AI on low code development platforms can be identified?

RQ 2: What future direction is the rise of generative AI in low lode development platforms leading to?

To answer the stated research questions, we conduct expert interviews from the field of low code to identify challenges and future directions for the impact of generative AI on low code. Through research and application of AI, we are constantly exploring what is possible in the future by taking available and plausible actions in the here and now (Berente et al., 2021).

This paper is structured as follows: Section 2 provides the conceptual background of this work. Section 3 presents the methodology of the applied research approach, data collection, and analysis. Section 4 presents the findings of the study. Section 5 provides a summary as well as limitations and potential future research directions.

2. Conceptual Background

This section provides additional context on Low Code Development Platforms, Generative AI (GenAI), and Artificial Intelligence Generated Content (AIGC).

2.1. Understanding Low Code Development Platforms

The term Low Code Development was originally coined by the market research company Forrester Research (Richardson et al., 2014). They characterized LCDPs as "*platforms that enable rapid delivery of business applications with a minimum of hand-coding and minimal upfront investment in setup, training, and deployment*" (p. 2). In 2017, Forrester Research offered a more detailed definition for LCDPs, describing them as "*products and/or cloud services for application development that employ visual, declarative techniques instead of programming and are available to customers at low-or no-cost in money and training time to begin, with costs rising in proportion of the business value of the platforms*" (p. 4).

In more detail, those platforms are IT artifacts that support the development of software, applications, and workflows using minimal source code and a graphical interface instead of complex programming languages (Al Alamin et al., 2021; Sanchis et al., 2019). As LCDPs are most of the times cloud-based systems, they can be integrated with existing workflows developed with popular Software-as-a-Service (SaaS) applications, such as Zapier, Amazon AppFlow, and Trello, to only name a few (Di Ruscio et al., 2022). The main objective of LCDPs is to reduce the development, deployment, and maintenance effort of software compared to the cost and effort of traditional software development.

Additionally, by enabling digital-savvy employees, so-called citizen developers, employees with limited programming knowledge or experience can also directly contribute to the software development process (Di Ruscio et al., 2022). Thus, LCDPs drive the empowerment of employees from across the organization to realize their ideas on LCDPs and thus be a part of the digital innovation process (Krejci et al., 2021).

When describing LCDPs, a special emphasis is being put on the WYSIWYG principle (acronym from What You See Is What You Get) which is a computing term that describes software that accurately reflects the appearance of the final product. In Low Code, the WYSIWYG-principle describes that everything is built visually, with assistance of a variety of well-built

templates to develop applications (Luo et al., 2021). In 2016, Gartner identified a similar segment called low-code application platforms (LCAP), characterizing it as a platform for development and deployment for any user experience, business process, business logic, and data (Vincent et al., 2019).

Nevertheless, we have to mention that low code development cannot be considered a new phenomenon as such (Krejci et al., 2021) and shows close parallels with existing research streams such as rapid application development (Ismail, 2017; Vincent et al., 2019) and model-driven engineering (MDE) , amongst others. These approaches broadly share common goals such as raising the abstraction level for software development and hiding implementation-level details (Di Ruscio et al., 2022).

However, LCDPs are distinct in their overarching aim to empower users across the organization to perform application development activities in their own, mostly non-IT departments. Moreover, di Ruscio et al. (2022) argue, that while the main goal of MDE is to increase productivity by automating different steps in software development, LCDPs promote the construction of applications using forms and graphical editors with little or no manual source code.

2.2 Conceptualizing Generative AI and Artificial Intelligence Generated Content

Generative artificial intelligence is a class of machine learning technologies that "generate new content" […] and include "text, images, music, or video by analyzing patterns in existing data (Brynjolfsson et al., 2023, p. 3-4). This paper focuses on GenAIs that are based on large language models (LLMs). On an abstract level, they are neural network models specialized to work with sequential data. These models were trained on large amounts of text data (a corpus) to predict the next most likely following word based on co-occurance of words (Bubeck et al., 2023).

Recent advances in generative AI were catalyzed by two model architectural advances of self-attention and positional encoding (Vaswani et al., 2017). Pre-training LLMs on large amounts of unlabeled data and further fine-tuned to create general purpose LLMs result in Generative Pre-trained Transformer (GPT) models, such as ChatGPT (Liu et al., 2023).

Artificial Intelligence Generated Content (AIGC) describes content that is generated using advanced Generative AI, such as GPT models. AIGC, allow the automated creation of large amounts of content in different media in a short amount of time (Cao et al., 2023). Such systems work in two steps (Liang et al., 2023; Stefanini et al., 2023): First, the system extracts intent information from human instructions to teach and guide the underlying model. Second, GenAI creates content according to the extracted intentions that satisfies the instructions (Cao et al., 2023). The core advancements in recent AIGC developments compared to prior techniques is the result of three underlying effects such as training more sophisticated models on larger datasets, using larger foundation model architectures, and having access to increased computational resources. Additionally, experts are also constantly exploring novel ways to integrate new technologies with GenAI algorithms (Cao et al., 2023).

2.3 Managing Artificial Intelligence (AI)

In their MISQ special issue Berente et al. (2021) identified three relevant concepts regarding managing AI since *"Managing AI involves communicating, leading, coordinating, and controlling an ever-evolving frontier of computational advancements that references human intelligence in addressing ever more complex decision-making problems. It means making decisions about three related, interdependent facets of AI—autonomy, learning, and inscrutability—in the ongoing quest to push the frontiers of performance and scope of AI"* (p.1).

Since GenAI is increasingly impacting system development and LCDPs and the way we develop new systems, the concept of the three identified frontiers helps us to analyse how the frontiers in system development will evolve.

In their paper, Berente et al. (2021) identified two dimensions of AI frontiers including the three facets autonomy, learning, and inscrutability. Performance relates two tasks that could be done by AI while the scope includes the application areas of AI. Within these dimensions autonomy describes acting without human intervention; learning improving through data and experience; and inscrutability being unintelligible to specific audiences.

3. Methodology

To answer our research questions, we rely on a qualitative research design using expert interviews following Babbie (2020) and Mayring (2015). Given the accelerated advancements in GenAI and Artificial Intelligence AIGC, there exists a temporal disconnect where research struggles to keep pace with practical implementations, particularly in the swift execution of innovative concepts and the evaluation of their use cases.

In total, we conducted ten in-depth interviews with LCDP experts who are either users of LCDPs, suppliers of LCDPs, or consulting companies that implement LCDPs. Table 1 provides additional information about the organization the interviewed experts are affiliated with (group, firm size, industry) and the expert's position.

I	Group	Firm Size	Industry	Position
1	Supplier	Small	IT-Services & IT-Consulting	Senior IT Architect
2	Supplier	Small	IT-Services & IT-Consulting	Co-Founder, CTO
3	Supplier	Small	IT-Services & IT-Consulting	Co-Founder and Head of Solutions
4	Supplier	Large	IT-Services & IT-Consulting	Technical Specialist Business Applica-tions
5	User	Large	Pharma-ceutics and Medicine	Junior IT Architect
6	User	Medium	IT-Consulting	Junior IT Architect
7	User	Medium	Pharma-ceuticals, Biotech-nology, and Chemicals	Head of IT
8	Consul-tant	Large	IT-Services & IT-Consulting	Associated Manager, Senior Low Code Designer
9	Consul-tant	Large	IT-Services & IT-Consulting	Head of Low Code
10	Consul-tant	Medium	IT-Services & IT-Consulting	Senior Manager

Table 1. Overview of Interview Participants.

The interviews were conducted via the video conferencing tool MS Teams and lasted between 27 and 52 minutes. The interviews were held in Swiss German, German, and English. We stopped conducting interviews as soon as we realized that no additional insights were revealed on the foundation of the theoretical saturation proposed by Glaser and Strauss (2017). Each interview was recorded and transcribed using the automated transcription plug-in of MS teams for English and German interviews and the transcription software töggl.ch for interviews conducted in Swiss German. The AI-generated transcripts were each checked for correctness and consistency and manually adapted when needed.

Subsequently, the interviews were coded by two of the authors. By following an open-coding approach, the content of the interviews was transferred into codes and subcodes. To ensure dependability and reliability, the coding process was conducted by the two authors independently. After each session, the codes were discussed to ensure consensus.

In a final step, we translated the codes, subcodes, and key passages verbatim quotation into English. In the following passage, the associated research outcomes are described.

4. Findings

In the following section, the identified elements that emerge in the integration of GenAI in LCDPs are positioned within the current standpoint. Subsequently, it is indicated in which direction they are likely to develop in the future and thus push the AI and human frontier.

4.1 Scope Frontier

The following part describes the current relationship and connection of executed tasks on LCDPs through collaborative AI and human labor. Specific attention is paid to code generation, code review, content generation, data analysis, testing and tasks within security and privacy. The described dimensions, categories, and facets of AI and humans respectively are shown in table 2.

4.1.1. Code Generation. Most of the interviewees agreed on the possibilities of automated code generation by GenAI in LCDPs: In the future, it will become more and more likely that GenAI will produce a large part of the code with little human input. This will not only improve existing code, produce a lot of code in a short period of time, but also provide use-case specific content and reduce the error-proneness of code due to human errors.

Thus, both coding novices and professional software developers can trigger the code generation of GenAI with just a few inputs and implement initial ideas while avoiding errors. Interviewee 6 described it as follows: *"(…) AI-generated code can certainly be a help and reduce work in the future, I mean we already have our own code editor that we use. There are also plug-ins that help with coding. This means that you enter something and then AI suggestions and*

recommendations for code generation and code completion come up. That's a big help and will be used more in the future."

However, interviewee 8 emphasized that automatic code generation is particularly useful for the development of small, independent applications. For applications that are part of a company-wide IT landscape, it is more difficult to solve this with automatic generated code due to the large interconnections and integration in other IT-systems. Or as he specifically stated: *"Some already think that developers can easily be replaced by the automatic generation of code by AI tools. But that is not the case at all. Especially when you're in the enterprise environment, you can't just do that. If you want to make a standalone app, you might be able a try existing system that's running productively and serving a lot of customers, you can't just take code and make it live in an application like that."*

Thus, in this context, one is and will be dependent on the knowledge and understanding of experienced IT professionals.

Furthermore, the "blackboxing" of automatic code generation was also mentioned as a key issue regarding the future development of LCDP and GenAI: Two interview participants explicitly mentioned the risk that IT professionals will experience more difficulties deciphering errors in AI-generated code.

"Since it is not possible to understand how the errors occurred and one cannot ask AI, this can make bug fixing more difficult or impossible". (Interviewee 4)

Therefore, in the future, it may be challenging if a large part of the code is generated automatically in the low-code environment.

4.1.2. Code Review. Similar to code generation, code review will also be heavily supported by AI-generated inputs. The interview participants also mentioned the fast processing of a lot of data and the low error rate for clear tasks as a major advantage of AI compared to human input. However, as with code generation, it is also important for IT experts to review the AI-generated input, and the review must not be accepted unquestioningly. Interviewee 1 predicted that AI will remain error-prone for quite some time and that human input will remain increasingly relevant.

However, as soon as the development of AI-models evolves further, it can be assumed that in the future there will also be certain automatically implemented consistency checks and best practices that AI can use to check generated code even more easily and accurately. Interviewee 8 also mentioned that in the future it is also possible that artificial intelligence could provide preventive assistance during code generation with verification and *"receive suggestions on what could happen next and check and compare this directly with the consistency and accuracy of the existing code".*

Eventually, interviewee 5 mentioned that the more complex the use cases, the more important human review of AI activities becomes. Therefore, especially in more complex use cases, close collaboration between AI and human input will be particularly important.

Dimensions of the frontier	**Category**	**Facets of AI**	**Facets of Human**
Scope Frontier	Code Generation	Reduce Human Errors	Reduce AI-Black-boxing
	Code Review	Automated Quality Assurance	Reviewed AI Code
	Content Generation	AI-generated content	Human-Generated Content
	Data Analysis	Unlimited Analyses	AI-assisted Data Analysis
	Testing	Automated Testing	Assisted Testing
	Security / Privacy	Automated Integration	Risk, Compliance Awareness
Performance Frontier	Development Quality	Improved Code Quality	Assured Code Accuracy
	Development Quantity	Automated Code	Customized Code

Table 2. Summary of Identified Dimensions of AI Frontiers.

4.1.3. Content Generation. Regarding content generation, GenAI-based inputs will also offer many opportunities in the Low-Code environment. Besides existing possibilities such as the AI-based generation of images, texts and videos, one interview partner especially empathized the possibility to generate a user interface based on a drawing or sketch. In this way, you can get an initial idea of what you can build on in the future with little time and money.

However, interviewee 7 mentioned an important objection: While it's very convenient and useful to

have GenAI-generated content as a starting point of an IT-artifact, GenAI can develop a lot of content, of which it is not clear what and how much of it is actually useful and meaningful: "So I imagine that in the very near future, we will all be drowning in lots of very nice well-written text that nobody wants to read" (interviewee 7).

Therefore, an important task for people in the future will be to remain responsible for (at least part of) the content generation and to increasingly check which content is purposeful and useful for the company and the further development of IT artifacts.

4.1.4. Data Analysis. Regarding data analysis, the interviewees particularly mentioned two opportunities of GenAI in connection with LCDPs: Firstly, AI-supported systems have the possibility to think a few steps ahead of the user and thus to point out possible opportunities and obstacles in the development of IT artifacts in a preventive way. Interviewee 7, for example, put it this way: *"I would find it advantageous if the AI-supported assistant would "work with me" and directly point me towards already existing content or linkages for example in other work products".*

On the other hand, GenAI has a huge amount of data available for analysis and can therefore quickly make an excerpt of data sets and make suggestions based on this essence (e.g., Interviewee 7).

With respect to the human resources of data analysis, the interviewed experts agree that GenAI has clear advantages over the human user. It can be assumed that this task will become even more AI-aided in the future.

4.1.5. Testing. The interview partners mentioned many opportunities as well as barriers in the area of test automation, and the topic can develop in many directions for GenAI in connection with LCDPs. Half of the interviewees mentioned that automated creation of test cases, execution of testing and subsequent documentation of test cases can take a lot of work off the shoulders of development teams. Especially in the low code environment, as this work falls back on the IT professionals due to a lack of knowledge and expertise in the testing area, despite the extension of the responsibility of the development activity outside the IT team.

In this context, one interviewee mentioned in particular that for him testing automation is not part of generative AI, but rather belongs to the area of general automation of workflows. *"(...) unless you have an AI that looks into the system itself, analyses it and produces test cases for you. Then for me it would be a generative AI that takes care of the testing for me."* (Interviewee 6)

4.1.6. Security / Privacy. Industry experts expressed concerns surrounded risks and compliance associated with GenAI in LCDPs. They stressed the ongoing need for risk management and compliance measures.

"(…) when an AI algorithm has access to sensitive data of individuals or companies, it must be ensured that this data is protected and used in accordance with applicable data protection laws. (Interviewee 9)

Thereby, they especially mentioned the importance of identifying, describing, and categorizing the different use cases where GenAI is used in LCDPs. Based on this categorization, a risk assessment and the corresponding compliance measures then must be initiated.

Another security risk can arise with AI from the training data used to train the AI. It requires understanding, managing, and mitigating the "data black box" that can arise around content in the areas of discrimination and bias or copyright infringement (Interviewee 10). Eventually, the challenges associated with risk management and compliance in LCDPs and GenAI do not fundamentally differ from the requirements in traditional development processes. However, one interviewee also empathized the *manageable nature* of these risks is largely due to the non-core nature of the systems currently being built using LCDPs: *"In Low Code, I would say, the risk is relatively manageable. Because these are not core systems that you develop entirely with Low Code Development Platforms."* (Interviewee 8)

4.2 Performance Frontier

4.2.1. Code Quality. Quality of code entails the need for refactoring the codebase, which can mean that the code is written in a way that causes inefficiency during runtime or that it is not easily maintainable. The industry experts see the potential and issues revolving GenAI and how to assure code quality. For some complex applications there are risks that the complex dependencies of software could negatively influence code quality for LCDP customizations.

4.2.2. Development Time. LCDP and GenAI can lead to reducing development time. With a skilled developer, testing different generated codes will become much quicker. If the developer has the necessary LCDP and domain knowledge to formulate the problem, GenAI can help in creating deployable source code and help with finding problems in existing

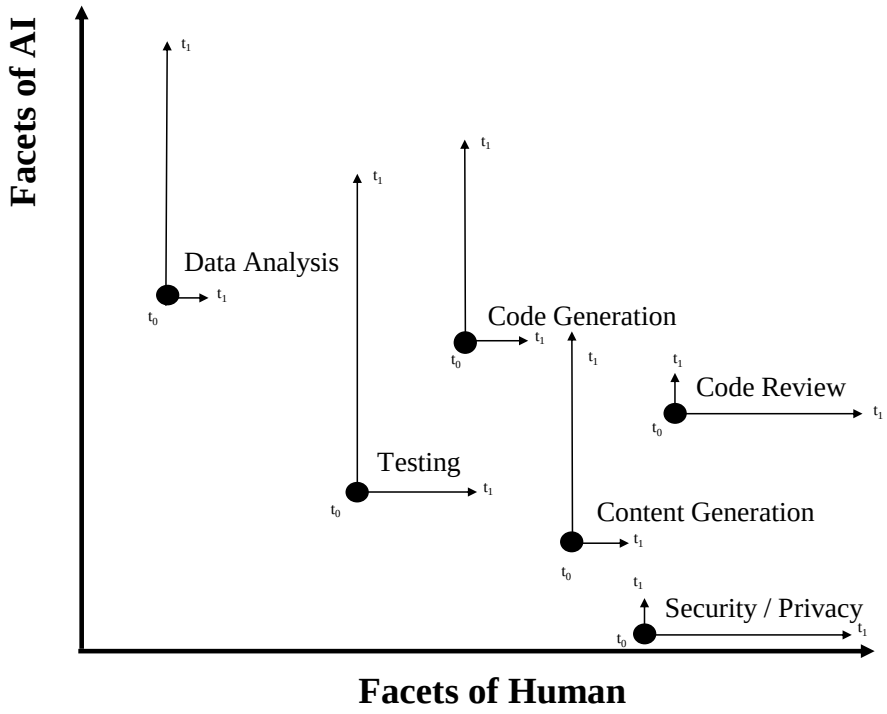

Figure 1. Future Directions of AI and Human Facets in Low Code Development.

source codes. This leads to shorter development and testing cycles.

4.3 Future Directions of Low Code Development

As part of this section, we discuss the contributions of our study and provide an agenda for future research directions. We see it as important to further advance the understanding of possible development scenarios of GenAI LCDPs, as well as their potential and challenge, that we have arrived at in the context of this study by further exploring seemingly contradictory findings, as well as by presenting our findings toward possible research. Figure 1 gives an overview of the identified development scenarios and points towards the future directions 1-6 that will be explained in this section.

The small amount of practice experience on generative AI in low code development as well as less literature on this topic demonstrates that this is still a new research area. 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 and new features resulting from generative AI. We divide our agenda for future research directions into six directions.

GenAI-Low Code Development has a major impact on system development in practice and research in the coming years, on the one hand bringing expertise from other disciplines into system development, but also bringing system development into entirely new areas.

When we asked the experts whether they considered low code development to be threatened by GenAI in the long term, the tendency pointed towards an interplay between human and AI-generated input in the future. A possible procedure is that AI can provide a person with an initial suggestion using just a few inputs. This can be used by experienced IT professionals or people with few IT skills and knowledge. Afterwards, however, it is of great importance to be equipped with the necessary IT skills and knowledge to assess whether the AI-generated input is useful for the realization of the desired IT-solution.

Our interviews indicate that a shift in citizen development will occur as soon as generative AI takes over tasks of the development process. So far, low code platforms are often used to empower people without programming skills, but if even these tasks are now taken away, it may lead to even less technical understanding being necessary. However, the danger arises that a lack of technical understanding can lead to programs being of poor quality and cause problems in the long term.

Interviewee 9 therefore sees the need to link citizen developer and pro developer even more closely

together with the possibilities of generative AI and calls this "*fusion development*". Thus, resulting in our first future direction:

Future Direction 1: Most of the code is written by the AI and finalized by the human.

At the present time, testing is often one of the unpopular and tedious tasks and could be relieved by automatic testing. As the shift towards automation and acceptance by generative AI continues, the burden on humans will be reduced. GenAI not only automates but also optimizes software testing processes. AI-based testing can potentially cover more ground, detect subtle defects, and handle complex test scenarios more efficiently than humans, thereby increasing software reliability and robustness.

This point was empathized by interviewee 3 for example: *"Testing can certainly be automated, as long as the test data corresponds to the actual facts and the human being then also checks it."*

The ongoing development of AI testing algorithms will further augment this automation. Which leads to our second future direction:

Future Direction 2: Testing of programs is fully automated and performed by AI.

Security and privacy are unavoidable requirements, especially in the European area, based on the GDPR, the AI act and digital service act (Barati et al., 2019). So far, this task is strongly in human hands and will remain so in the future. It may even become more of a human task during development if other tasks fall away. While AI can help identify security risks or breaches in privacy, the responsibility of implementing and maintaining secure, privacy-compliant systems will remain a human task. Interviewee 5 compared this fact to the present situation as follows: *"When generating code with AI, you just have to make sure that the functions that are provided are safe and reliable. That means, for me, there is not much difference to what we have today."*

Thus, the critical nature of these tasks, exacerbated by regulations such as GDPR and the AI Act, requires the use of human being to interpret and apply these guidelines contextually. Thus, resulting in our third future direction:

Future Direction 3: Security and privacy considerations is ensured by the human being.

At the present time, AI is already far superior to humans in content generation. For example, through AI-based generation of images, texts and videos. It is to be expected that this will continue in the direction of AI in the future, with little human involvement. The ability of AI to generate content will be expanded to more complex and creative domains, such as designing user interfaces or creating multimedia content for applications. However, final approval and contextual fitting will continue to be a human responsibility to ensure alignment with business objectives and audience needs. Our fourth future direction therefore relates to the content generation:

Future Direction 4: Changing the way of content generation.

Due to new possibilities to analyze and improve data, data analysis in and with low code development platforms will improve in the future. These tasks will lie less with humans themselves and will be primarily performed by AI. Humans will be able to focus on other tasks and will not have to spend time analyzing data. Interviewee 4 summarized it as follows: *"AI is simply superior to us in content creation. it can create so much in so little time. we as humans simply have to make sure that the content makes sense to me."*

This results in future direction five:

Future Direction 5: Data analysis will be improved due generative AI.

Code review, on the other hand, builds on human skills and requires human contextual assessment and analysis. Even with AI support, code review will nonetheless demand human judgment and interpretation. AI can help by detecting potential problems and enforcing coding standards. Understanding the big picture, including the business context and goals, will, however, remain a uniquely human ability. Therefore, it comes to the following assumption in research direction six:

Future Direction 6: Code review tasks builds on human skills.

5. Discussion, Limitations, and Future Research

Our data allow us to make assumptions about possible effects of generative AI and how it shapes the future of low code development. In addition to announcements from various platform providers, there is a lack of practical experience with the low code platforms and generative AI features. However, because low code itself is undergoing many different changes and is constantly changing, we provide initial research directions since low code development has already had a major impact on companies and the way they develop systems.

Our study has limitations and offers room for further research. In order to get a comprehensive perspective on the influence of generative AI on low code development more data and especially more experience with this tool are necessary. In addition to a relatively small interview participant sample, we spoke with active low code users in our interviews. Future research should conduct more interviews as well as interview low code platform providers about their planned changes.

The interviews already show that low code has generally changed the world of work, has become integrated into companies and will continue to be adapted.

6. Conclusion

This research provides a nuanced understanding of Generative AI (GenAI) in Low Code Development Platforms (LCDPs), using expert interviews to investigate the intersection of AI and human tasks in software development. According to our experts, GenAI has a lot of potential for tasks like automated code generation, code review, content generation, data analysis, testing, and certain aspects of security and privacy. This potential, however, is dependent on a number of factors, including the complexity of the IT landscape and the level of human oversight.

In conclusion, incorporating GenAI into LCDPs offers promising opportunities for accelerating tasks, increasing efficiency, and decreasing errors. Nonetheless, the study emphasizes the importance of preserving a balanced, symbiotic relationship between AI and human roles, emphasizing the continued relevance of human expertise in code review, content generation, and other areas. As we progress into this new era of AI-driven system development, both researchers and practitioners must navigate this dynamic landscape while taking into account both the opportunities and challenges it presents.

7. References

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

Babbie, Earl. (2020). The practice of social research. Fifteenth Edition. Bd. CENGAGE. Chapman University.

Berente, Nicholas, Gu, B., Recker J., & Santanam, R. (2021). Managing Artificial Intelligence. In *MIS quarterly*. 45(3).

Bouschery, S. G., Blazevic, V., & Piller, F. T. (2023). Augmenting human innovation teams with artificial intelligence: Exploring transformer-based language models. In *Journal of Product Innovation Management*. 40(2), 139-153.

Brynjolfsson, E., Li, D., & Raymond, L. R. (2023). Generative AI at work (No. w31161). In *National Bureau of Economic Research.*

Bubeck, S., Chandrasekaran V., Eldan R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y. & Lundberg, S. (2023). Sparks of artificial general intelligence: Early experiments with gpt-4. In *arXiv preprint arXiv:2303.12712.*

Bucaioni, A., Cicchetti, A., & Ciccozzi, F. (2022). Modelling in low-code development: a multi-vocal systematic review. *In Software and Systems Modeling*. 21(5), 1959-1981.

Cao, Y., Li, S., Liu, Y., Yan, Z., Dai, Y., Yu, P. S., & Sun, L. (2023). A comprehensive survey of ai-generated content (aigc): A history of generative ai from gan to chatgpt. In *arXiv preprint arXiv:2303.04226*.

Dang, H., Mecke, L., Lehmann, F., Goller, S., & Buschek, D. (2022). How to prompt? Opportunities and challenges of zero-and few-shot learning for human-AI interaction in creative applications of generative models. In *arXiv preprint arXiv:2209.01390.*

Di Ruscio, D., Kolovos, D. de Lara, J. Pierantonio, A. Tisi, M. & Wimmer, M. (2022). Low-code development and model-driven engineering: Two sides of the same coin? In *Software and Systems Modeling*, 21(2), 437–46.

Di Sipio, C., Di Ruscio, D. & Nguyen, P. T. (2020). Democratizing the Development of Recommender Systems by Means of Low-Code Platforms. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, 1–9.

Dwivedi, Y. K., Kshetri, N., Hughes, L., Slade, L. Jeyaraj, A., Kar, A. K., Baabdullah, A. M., u. a. (2019). Opinion Paper: "So What If ChatGPT Wrote It?" Multidisciplinary Perspectives on Opportunities,

Challenges and Implications of Generative Conversational AI for Research, Practice and Policy". In *International Journal of Information Management* 71: 102642.

Glaser, B. G., and Strauss, A. L. (2017). Discovery of grounded theory: Strategies for qualitative research. *Routledge*

Harris-Watson, A. M.., Larson, L. E., Lauharatanahirun, N. DeChurch, A. L. & Contractor, N. S. (2023). Social Perception in Human-AI Teams: Warmth and Competence Predict Receptivity to AI Teammates. *In Computers in Human Behavior 145*.

Tyson, K., Wu, L. & Aycock, J. (2023). AI-Generated Code Not Considered Harmful. In *Proceedings of the 25th Western Canadian Conference on Computing Education*, 1–7. Vancouver BC Canada: ACM, v 2023.

Khorram, F., Mottu, J.-M., & Sunyé, G. (2020). Challenges & Opportunities in Low-Code Testing. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, 1–10. Virtual Event Canada: ACM, 2020.

Krejci, D., Iho, S. & Missonier, S. (2021). Innovating with employees: an exploratory study of idea development on low-code development platforms. In *European Conference in Information Systems* ECIS 2021 Research Papers. 118.

Liang, P. P., Zadeh, A. & Morency, L.-P. (2023). Foundations and Trends in Multimodal Machine Learning: Principles, Challenges, and Open Questions. In *arXiv, 20*.

Liu, Y., Han, T., Ma, S., Zhang, J., Yang, Y., Tian, J., ... & Ge, B. (2023). Summary of chatgpt/gpt-4 research and perspective towards the future of large language models. In *arXiv preprint arXiv:2304.01852*

Luo, Y., Liang, P., Wang, C., Shahin, M., & Zhan, J. (2021). Characteristics and challenges of low-code development: the practitioners' perspective. In Proceedings of the 15th ACM/IEEE international symposium on empirical software engineering and measurement (ESEM) (pp. 1-11).

Mayring, P. (2004). Qualitative content analysis. A companion to qualitative research, 1(2), 159-176

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

Richardson, C., Rymer, J. R., Mines, C. Cullen, A. & Whittaker, D. (2016). New development platforms emerge for customer-facing applications. In *Forrester Research*.

Sanchis, R., García-Perales, O. Fraile, F. & Poler, P. (2019). Low-Code as Enabler of Digital Transformation in Manufacturing Industry. In *Applied Sciences* 10, 1.

Stefanini, M., Cornia, M., Baraldi, L. Cascianelli, S., Fiameni,, G. & Cucchiara, R. (2023). From Show to Tell: A Survey on Deep Learning-Based Image Captioning. In *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, Nr. 1 (1. Januar 2023): 539–59.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing system*s, 30.

Vincent, P., K. Iijima, M. Driver, W. Jason, und Y. Natis. (2019). Magic Quadrant for Enterprise Low-Code Application Platforms. In *Gartner*.