Impact of No-Code Technology on Startup Founding

Florian Metzger

Dissertation written under the supervision of Prof. Renzo Cenciarini

Dissertation submitted in partial fulfilment of requirements for the International MSc in Management with Major in Entrepreneurship and Innovation, at Universidade Católica Portuguesa and for the MSc in Economics and Management of Innovation and Technology at Bocconi University, 3rd of January 2024.

Abstract

The purpose of this study is to understand the implications of No-Code technology on startups. For this purpose, the drivers and inhibitors of No-Code for startup founding and the implication of No-Code solutions on the composition of the startup team was examined. A group of 12 participants among No-Code experts and investors took part in a semi-structured interview with questions relating to experience, usage, skills and investment criteria.

The greatest value and driver of No-Code solutions lies in the initial phase of startups, as they enable founders to quickly develop and validate their ideas with limited resources. For simple applications and validation, No-Code is considered state-of-the-art in software development. In advanced stages of the startup development as the complexity of the tasks increases, it often becomes necessary to transition to custom code. Lack of trust in the technology and fear of becoming dependent on a vendor dependency are the main reasons noted on why to reject No-Code solutions.

When used, No-Code technology has an important impact on the composition of a startup team. Technical skills are only needed at a later stage and thus founders are not reliant on technical expertise in the initial phase. This empowers non-technical members to expand their role in the startup foundation. In the long term, a decreased demand for software developers is expected due to the increased efficiency, resulting from the combination of No-Code and AI. This addresses a currently pressing issue of missing software developers inhibiting software startups.

Keywords: No-Code, Low-Code, low-code development platform, LCNC, adoption theories, software development, initial team, founding team, competencies, software startup

Title: Impact of No-Code Technology on Startup Founding

Author: Florian Metzger

Abstrato

Este estudo visa compreender as implicações da tecnologia No-Code nas empresas Startups. Neste âmbito, foram examinados fatores impulsionadores e inibidores, bem como o impacto das soluções No-Code na composição de uma equipa. Um grupo de 12 participantes, entre especialistas em No-Code e investidores, participou numa entrevista semiestruturada focada na experiência, utilização, competências e critérios de investimento.

O benefício e a conduta das soluções No-Code sentem-se na fase de arranque das Startup, pois permite desenvolver e validar planos rapidamente, com recursos limitados. Para aplicações simples e validação, No-Code é considerado topo de gama no desenvolvimento de software. Numa fase avançada do desenvolvimento da Startup, à medida que as tarefas complicam, torna-se frequentemente necessário transitar para o código personalizado. O ceticismo na tecnologia e a dependência de um fornecedor são relevantes na rejeição das soluções No-Code.

As competências técnicas apenas serão necessárias numa fase posterior. Os fundadores da Startup não dependem de conhecimentos técnicos na fase de arranque. Isto permite que a equipa não-técnica alargue o seu papel na fundação da empresa nesta fase. A longo prazo, espera-se uma diminuição da procura de programadores de software devido ao aumento da eficiência, resultado da combinação de No-Code com Inteligência Artificial. Este tópico aborda a questão atualmente urgente da falta de programadores de software que inibe as Startups do ramo.

Palavras-chave: No-Code, Low-Code, plataforma de desenvolvimento low-code, LCNC, teorias de adoção, desenvolvimento de software, equipa inicial, equipa fundadora, competências, startup de software

Título: Impacto da tecnologia No-Code na fundação de uma Startup

Autor: Florian Metzger

Contents

List of figures

List of tables

List of abbreviations

GDPR	General Data Protection Regulation
GPT	Generative Pre-Trained Transformer
LCDP	Low-Code Development Platform
LCNC	Low-Code No-Code
LLM	Large Language Model
MVP	Minimal Viable Product

1 No-Code in the context of startups

1.1 Problem Statement

There is a shortage of qualified professionals in the field of software engineering and development, as too few software engineers and developers have the most sought-after skills to a sufficient degree. This shortage of trained professionals is increasing due to the rapid development of new technologies (Pham, 2021). According to a study by iCIMS, it takes over 50% longer to fill technical positions (Koster, 2019). Furthermore, iCIMS analysed over a three-year period, that only 60 percent of the tech staff in demand is recruited (iCIMS, 2019). Established tech giants with large HR budgets, such as Apple or Google, have an advantage over smaller companies when it comes to attracting and recruiting top tech-talent (Pham, 2021). The significant issue is the disparity between the increasing demand for proficient developers and the limited availability of these competent developers (Krajewski, 2021). The Entrepreneur has released an article suggesting that Low-Code/No-Code Platforms have the potential to address the shortage of engineers and developers by enabling employees to become Citizen Developers (Santalo, 2021).

1.2 Significance of topic & research purpose

Dushnitsky & Straube (2021) examined the example of Shopify, and how the use of the Low-Code tools in venture-funded startups affects the allocation of human and financial resources. The advantage of Shopify is that it allows people without programming skills to run an online shop. The study shows that startups using Shopify have on average, a lower enterprise valuation at the outset, but also require fewer employees, resulting in lower expenses. The authors conclude that companies that use Shopify as an e-commerce platform need less capital and employees at the beginning but are later similarly successful as their competitors that use self-developed e-commerce platforms.

Traditionally, graduates from a master's in business administration (MBA) are often potential entrepreneurs and have the business knowledge to launch a startup. However, they often do not have the technical skills necessary to implement their business idea. Therefore, in the past, many founders either sought a co-founder with a technical background, or contracted the technical development externally, which is very costly. The use of Low-Code/No-Code tools can solve this dilemma, but raises new questions such as: how does the use of Low-Code/No-Code affect the internal organization and especially the composition of the founding team?

(Dushnitsky & Stroube, 2021) Luke Smith, Partner at the Early-Stage venture capital firm Forward Partners predicts an increasing number of ventures being built entirely with No-Code solutions and a considerable portion of them will receive venture capital funding (Smith, 2019). Will No-Code solutions make traditional coding respectively software developers redundant?

1.3 Research Approach

1.3.1 Objective

In the context of this paper, the implications of No-Code technology on the workforce formation of startups will be examined in further depth. The objective of this work is to understand the capabilities and limitations of No-Code to assist future founders in making decisions about the makeup of their founding team. This research also aims to investigate whether the exclusive use of No-Code tools has a negative impact on investment readiness. The findings may also be beneficial to No-Code platform vendors, allowing them to alter their solutions to better meet the needs of startups. Based on an in-depth literature review, provided in chapter 2, two research questions were developed. They fill an identified research gap and thus make a valuable contribution to the current research literature.

1.3.2 Research Question

RQ1: Which factors influence the utilisation and acceptance of No-Code solutions?

RQ2: How does the existence of No-Code solutions change the team composition of startups in different stages?

1.4 Thesis Structure

This thesis first, aims to introduce the reader to the current state of the art literature and its development regarding the No-Code market. In doing so, the existing advantages and disadvantages of the traditional software development on one side and the No-Code development on the other hand, will be elaborated. Furthermore, the impact on the skill-oriented composition of the founding team is shown. By strictly following the research methods, provided in chapter 3, the author demonstrates in the analysis in chapter 4 of the thesis, the structured findings, gained by applying the methods. In the discussion part (chapter 5), the empirical findings from chapter 4 will be placed in the context of the literature. Finally, the present thesis closes with the conclusion, the limitations of the research conducted and mentions further directions of research, that is needed to be carried out.

2 Literature review of Low-Code/No-Code in the context of startups

To acquire in-depth knowledge about software development and in particular about the No-Code industry and to be aware of the state-of-the-art literature, the following databases IEEE Xplore, ACM, Science Direct, EBSCO Host, Emerald Insight, SAGE, Springer, Google Scholar, JSTOR were searched mainly utilising the keywords: No-Code, low-code, low-code development platform, LCNC, adoption theories, software development, initial team, founding team, competencies, software startup. The purpose of a literature review is to provide the reader with an objective report of the existing literature to date on a particular topic (Ryan et al., 2007).

2.1 Startups

2.1.1 Startups and startup stages

Definition Startup

Giardino et al. (2014, p. 28) show a definition of a startup terming it as

> *[…] a small company exploring new business opportunities, working to solve a problem where the solution isn't well known and the market is highly volatile. Being newly founded does not in itself make a company a startup. High uncertainty and rapid evolution are the two key characteristics for startups […], which better differentiate them from more established companies.*

Within this thesis, the definition provided, above, will be taken into account. Also, the status quo of a startup is constantly changing, passing several stages of maturity (Alemany & Andreoli, 2018), which can be seen in the subsequent chapter.

Startup Stages

Several classifications about the stages of entrepreneurial ventures exist. This work uses the same classification used by Alemany & Andreoli (2018). The initial phase, known as the seed phase, involves testing and validating an existing idea in order to acquire first paying customers. The duration of this phase is dependent on the type of entrepreneurial venture. An entrepreneurial venture enters the startup phase when there is demand for its product or solution, already has customers, but expenses still exceed revenues. Once break-even is reached, the venture enters growth stage. The maturity stage is reached, as soon as market growth returns to normal levels (Alemany & Andreoli, 2018).

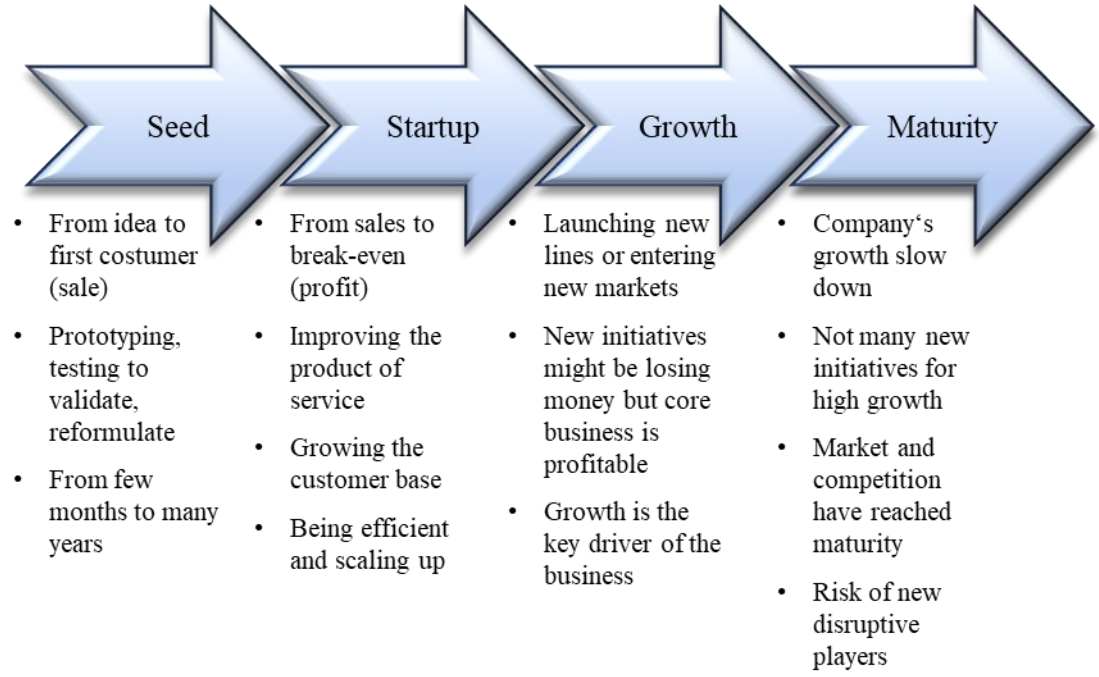

Figure 1: Different stages of venture development
Source: Adapted from Alemany & Andreoli *(2018, p.13)*

2.1.2 Key challenges of startups

After analysing causes of 111 startup failures, CB Insights has compiled a list of the most frequent cited reasons for failure. The most common mentioned, with a frequency of 38%, is running out of money and not being able to secure funding. The second most common cause is no market need, with a frequency of 35%. At the seventh place with 14% of occurrence, a wrong team composition is mentioned (CB Insights, 2021). The utilisation of No-Code solutions can help entrepreneurs to assess market need, by building a Minimal Viable Product (MVP), thereby testing demand, and receiving customer feedback at minimal expenditure (Smith, 2019). However, Giordano et al. (2015) point out that navigating the increasing technical unpredictability and obtaining the first paying client are the two largest obstacles software startups are facing in their initial years. The authors criticize that the learning process is not given nearly enough consideration while developing software. Crowne (2002) claims that signs of unskilled developers include failing to fulfil deadlines for product delivery, providing unreliable solutions, or taking a long time to rectify errors. Unexperienced software developers disregard non-coding aspects including *"[...] architecture, design, testing, configuration management, deployment, and documentation"* (Crowne, 2002, p. 338). Due to low budgets, or hardly sufficient capital in the early stages, startups resort to less experienced employees. Indicators of an unreliable solution are frequent and significant bugs that can only be resolved by developers with extensive experience. One of the factors causing such issue is a premature

release, which can result in serious consequences including reputational damage. The delay of the implementation of new functionality is the outcome (Crowne, 2002). According to him, to counteract the bottleneck of specialists, knowledge can be transferred from experts to new hires to distribute the required knowledge or skills among several people.

Giardino et al. (2014) have compiled a list including common challenging themes startups are facing: limited resources, high degree of reactivity, uncertainty, rapid evolvement, constant time pressure, third-party-dependency, small team and team members with limited experience, no organizational culture at the beginning, only one product or service, novel business, founder-centric, no upper management and high responsibilities, extremely risky, not profitable particularly in the early stage, and therefore require external funding. Giardino et al. (2015) have classified these challenges into four dimensions: product, market, financial, and team. Creating a product with features that customers value, understand the market and need, having enough financial resources to operate and put together a founding team that is able to master the above-mentioned challenges.

2.1.3 Core competencies of founding team of software startups

According to Blank (2011) a strong founding team is more important than a potential business idea itself. The core team should include four skills: 1) being a visionary, 2) possess design skills, 3) being a hacker and 4) having a hustler mentality. Since this is a great variety of skills and an individual usually does not have all competencies, a startup needs a team of people to cover those four core competencies. This is also a reason why the majority of startups are not founded by a single individual (Blank, S., 2011). Visionary in that context means to have the long-term goals in mind. The design capabilities allow to connect user and technology. This refers primarily to user experience and user interface expertise. The hacker skills in this context means the technological skills. Having the competence to build a solution or product. The fourth competence is to have a hustler mentality. In this context a generalist who is able to take responsibility in all areas. A person dedicated to taking responsibility and getting things done (Ganor, 2022).

Seppänen et al. (2016) divides the core team of a software startup into three categories, founder, experts, and team. Experts cover the skills of both founders and the team but can be rather assigned to the latter. With the founder requiring an innovation competence and the experts and team being responsible for its implementation. Those two competencies address two out of the four challenges software startups are facing introduced in chapter 2.1.2. The innovation

competency includes financing and the vision of the idea, while implementation includes e.g. software development, which belongs to the product dimension.

2.2 Theories and models employed for technology adoption

An adoption process is *"the mental process an individual passes from first hearing about an innovation to final adoption"* according to Rogers (1962) (as cited in Feder et al. (1985, p. 256)). *"[A]doption of technology is a process that leads to usage of a new technology, and might spread to other people through the diffusion process"* (Dissanayake et al., 2022, p. 2). The diffusion process regarding a special technology is *"[t]he process by which an innovation is communicated through certain channels over time among the members of a social system"* (Rogers, 1983, p. 5). The literature provides several models and theories regarding factors influencing this adoption process, as summarized in Dissanayake et al. (2022): the (1) *Model for Innovation-Decision Process*, (2) *Transtheoretical Model*, (3) *Theory of Reasoned Action* or (4) *Theory of Diffusion of Innovation* shall be mentioned for instance. This thesis applies number one and four, which will be explained in the subsequent chapter.

2.2.1 Innovation-Decision Process Model

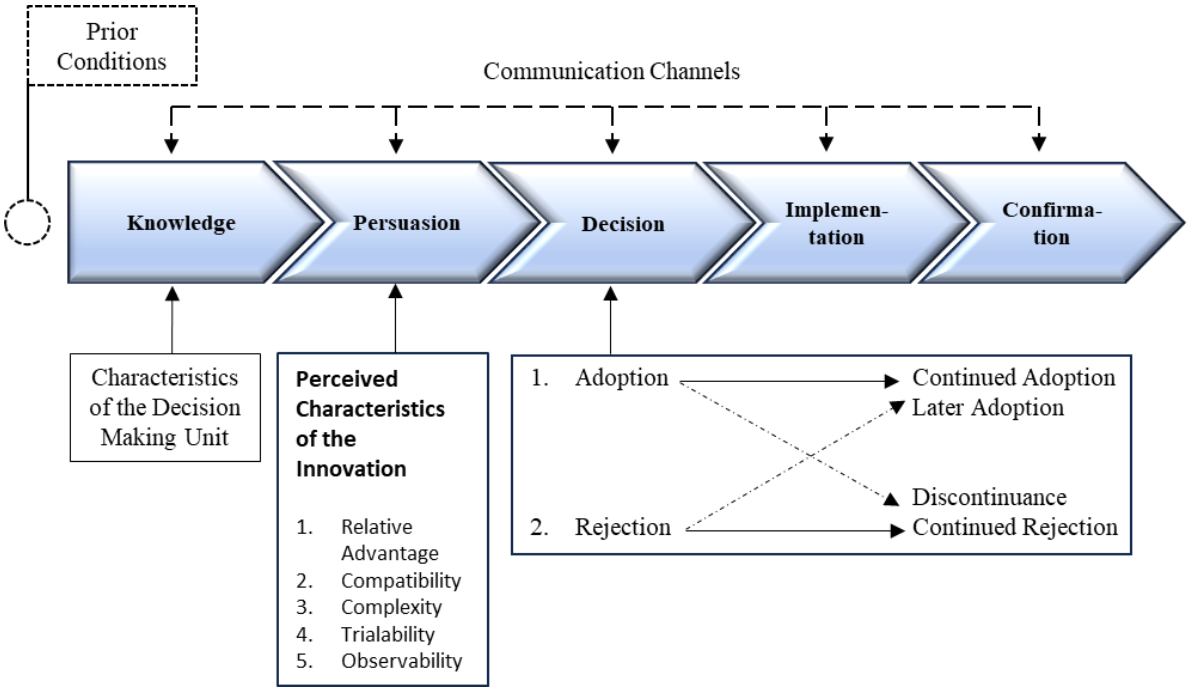

Figure 2: A model of five stages in the Innovation-Decision Process
Source: Adapted from Rogers *(2003, p. 108)*

The innovation decision process introduced by Rogers (1983) is comprised of five distinct stages. The initial phase is referred to as the knowledge stage, during which a particular person

becomes aware of the existence of an innovation. The awareness of an innovation might prompt to create a demand, creating a reciprocal relationship between innovation and needs. The acquisition of knowledge has the potential to generate an increased propensity to further explore the innovation. The second phase is known as the persuasion stage, during which the decision maker develops a positive or negative attitude. A description of the five characteristics of innovations which support the decision or rejection process, are displayed in the table below.

Characteristics of the innovation	
Relative Advantage	The extent to which an innovation is seen as inferior to the idea it replaces.
Compatibility	The extent to which an innovation is aligned with prior experiences, established values, or demands of its prospective consumers.
Complexity	The extent to which an innovation is seen as challenging to utilise.
Trialability	The extent wo which an innovation can undergo testing prior to being adopted.
Observability	The extent to which the outcomes are perceptible to observers.

Table 1: Characteristics of the innovation variables of persuasion stage
Source: Adopted from (Rogers, 1983; Rogers et al., 2019)

The third stage, known as the decision stage, occurs when the decision-making person decides in favour or against the innovation. The fourth stage of the innovation process is characterized by the implementation of the innovation and therefore known as implementation stage. The confirmation phase is the fifth phase. The decision-makers confirm the previously made decision in favour of the innovation or they reverse the decision if it has not led to the hoped-for success (Rogers, 1983).

2.2.2 Theory of Diffusion of Innovation

Rogers (1983) theory of Diffusion Innovation describes the Adoption on the one hand in form of a cumulative s-shaped-curve and on the other hand as a frequency based, bell-shaped curve with normality distribution. The latter can be horizontally divided into the five categories of adopters: Innovators, Early Adopters, Early Majority, Late Majority and Laggards.

Individuals within the group of innovators possess the ability to deal with a high level of uncertainty regarding the innovation, in other words, being venturesome. Early Adopters can be seen as role models for other participants within the social system, who get checked out by individuals before a new technology is used. This group helps to decrease the uncertainty. Members of the Early Majority are a little more cautious than their predecessors and rely on their judgement but are willing to adopt innovations. Late Majority individuals are very risk-averse and thus adopt innovations just after almost anyone has adopted it. However, Laggards tend to resist innovations and at a time they adopt an innovation, it's likely that already a new innovative idea is in place (Rogers, 1983).

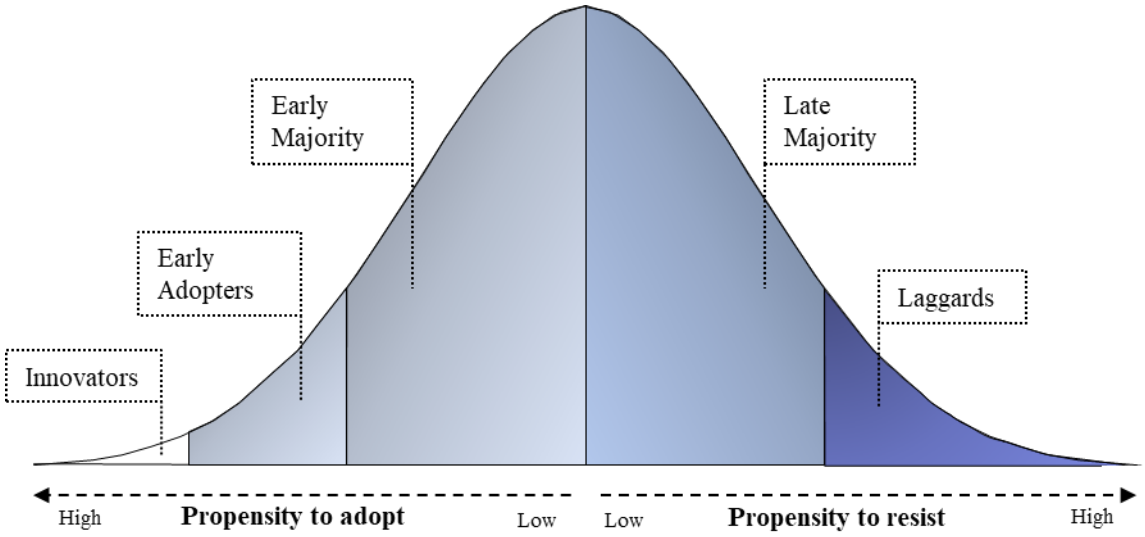

Figure 3: Adopter categorization of innovativeness
Source: Adapted from Rogers (1983, p. 247)

2.3 Emergence of the No-Code market

2.3.1 From traditional software development to No-Code

Over time programming languages evolved. O'Regan (2021) lists five generations of programming languages while other sources (Computer Science Degree Hub, 2022; Techbaz, 2023) mention six generations:

The first generation (1GL) also known as machine code, instructs computers using binary code. Creating a program with machine code can be challenging as it is very prone to errors, and these are difficult to correct (O'Regan, 2021). The first computer ever built in 1943 was able to understand binary, machine written codes (Computer History Museum, 2023).

The second generation language (2GL) is known as assembly language. In contrast to machine code, assembly language is simpler to read. 1GL and 2GL belong to the category of low-level language (O'Regan, 2021). The next progress took place in 1949, when assembly languages made it possible to set-up a machine-code in a simplified manner. With Autocode, an invention from 1952, programmers could use low-level programming language to generate algorithms, which afterwards were getting converted into machine codes (HP, 2018).

The third generation languages (3GL) belong to the category high-level language and were developed to facilitate it for humans to comprehend. They include languages such as FORTRAN, Pascal, C or COBOL. With high-level procedural languages, such as Algol or COBOL, it became ways easier to generate algorithms, since therefore, a smaller amount of code lines were needed (HP, 2018; Rizwan, 2018). High-level languages have for instance the benefits of an adequately defined syntax, readability, speed and suitability for scientific or business applications. 3GL can once again be divided in procedural an object-oriented language (O'Regan, 2021). Object oriented programming languages like Smalltalk were introduced around 1965 (Sufi, 2023). High-level programming languages are intentionally created to provide a level of abstraction to reduce unnecessary complexity, allowing programmers to concentrate on the actual task (Frampton et al., 2009).

The fourth generation languages (4GL) minimize programming effort and focus more on what must be done instead of how. One drawback of 4GL is that it is slower than the previous ones (O'Regan, 2021). 4GL for instance support web development projects and graphical user interface. This generation of technology enables non-technical people to build applications through codeless tools (Jurkenas, 2023).

The fifth generation languages (5GL) are primarily used in academia for research purposes, particularly in the domain of artificial intelligence. They are intended to direct computers to solve a problem, instead of the programmer (O'Regan, 2021).

The sixth generation language (6GL) is based on visuals. No-Code can be assigned to this generation (Computer Science Degree Hub, 2022; Techbaz, 2023).

Figure 4 below illustrates the chronological sequence of development.

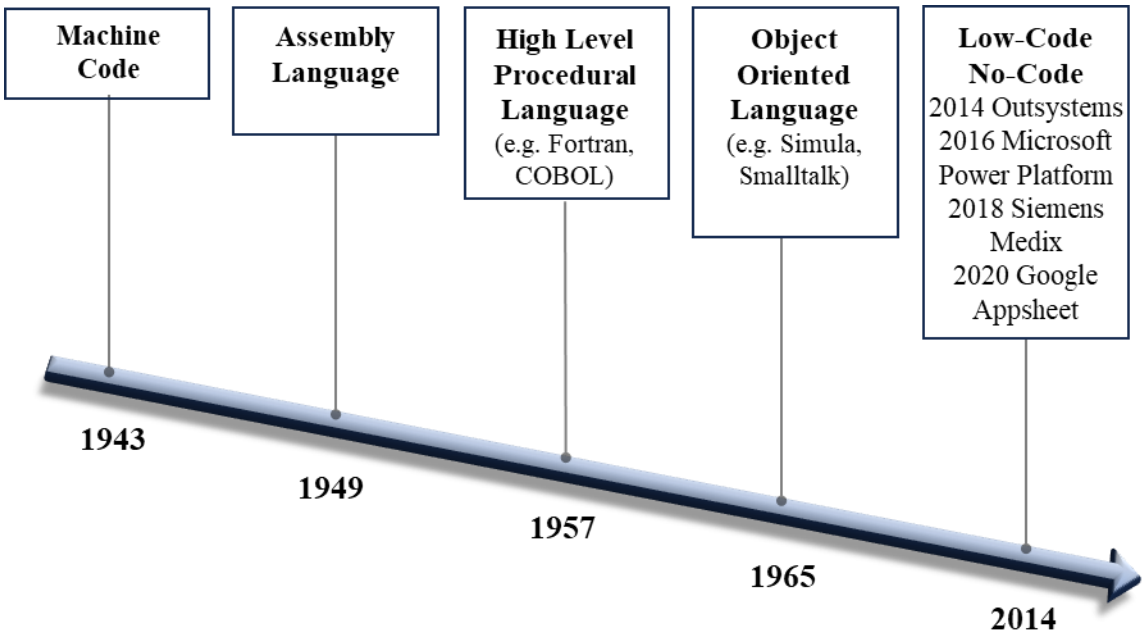

Figure 4: Evolution from Machine Code to No-Code
Source: Adapted from Sufi *(2023, p. 2)*

2.3.2 No-Code and Low-Code platforms

No-Code is the generation of visual software utilising so-called No-Code platforms and tools (Yarchevsky, 2021). No-Code eliminates the traditional programming procedure and enables visual software development (Adkin, 2020). There are several definitions of the terms of Low-Code and No-Code. According to the market researcher Gartner, a No-Code application platform is a subtype of a Low-Code application platform that solely needs text input and is therefore not categorized separately (Gartner, 2022). Low-Code development platforms can be defined according to the market researcher Forrester 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* (Richardson et al., 2014, p. 438).

Both Low-Code and No-Code offer the advantages including no reliance on developers and engineers, enhancement of productivity, faster customer feedback, possibility of customization compared to commercially available options, consistency, and cost efficiency (IBM Cloud Education, 2022). Nevertheless, there are differences which should be considered. The approaches are aimed at different target groups and qualification levels. Low-Code on the one hand is mainly aimed at developers or people with a general understanding to save time in standard development, so that those specialists can focus on innovative or new features. No-Code on the other hand is suitable for people without programming skills (IBM Cloud Education, 2022). In the interest of simplicity, the term No-Code is used throughout this study.

2.3.3 No-Code and Low-Code market overview

The total revenue for the Low-Code market in 2022 is approximately $22,5 billion and is expected to rise by 19,6% to $26,9 billion in 2023, and over $32 billion US dollars by 2024 (Gartner, 2022). Gartner (2022) conjoins Low-Code and No-Code application platforms together in its forecasts. In a study exploring the usage of No-Code tools, 38% stated that they would use them to *"Prototyping a new concept or product*", 19% to *"Automating processes or workflows"*, and 17% to *"Building an application"* (Formstack, 2021). There are different areas of application and various providers on the market offering Low-Code and No-Code solutions.

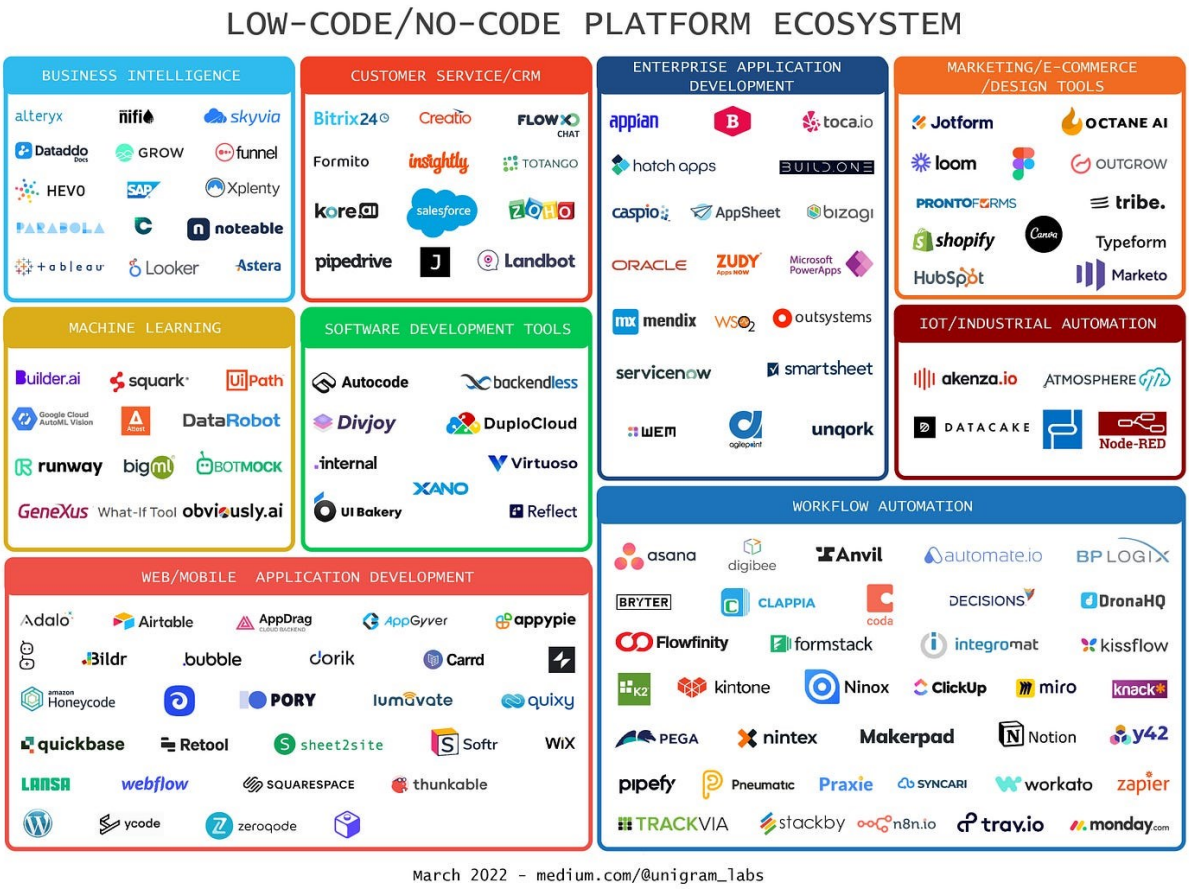

Figure 5: Overview of Low-Code/No-Code platform ecosystem

Source: From Unigram Labs (2022)

2.3.4 Development lifecycle approaches: Traditional vs. No-Code

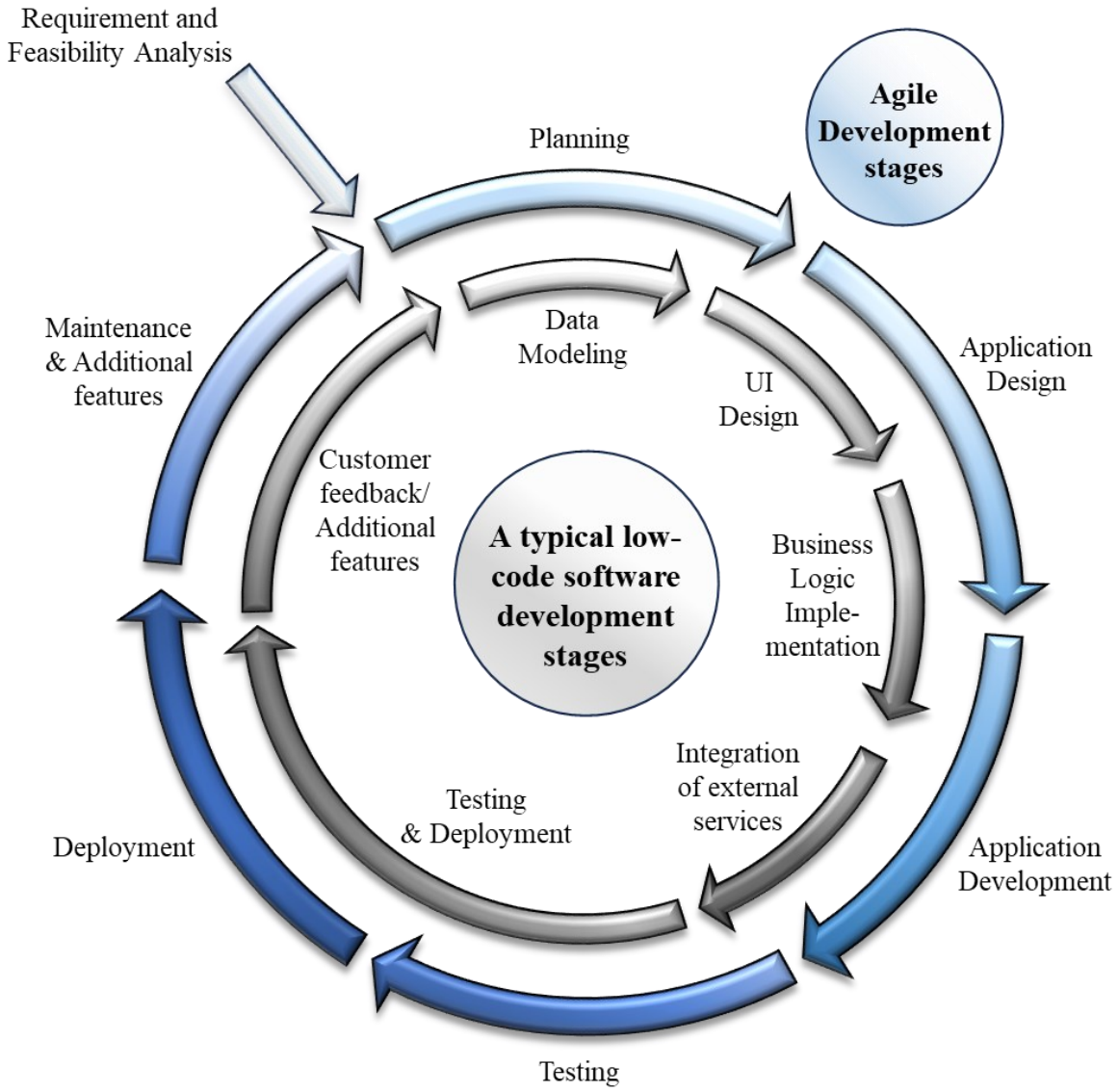

Figure 6: Agile versus Low-Code/No-Code software development
Source: Adapted from al Alamin et al. (2021, p. 47)

The grey coloured circle represents the Low-Code/No-Code software development stages (Sahay et al., 2020). According to the authors the first step in the Low-Code/No-Code is data modeling, where a data schema is configurated utilising drag and drop features. The second step is the user interface design. Here, the drag-and-drop function is essential in order to accelerating development. In the third step, business logic is implemented in form of visually supported workflows. In the following step, external interfaces are connected so that external data can also be processed. Most vendors provide a preview option and the possibility to deploy the application directly (Sahay et al., 2020). Whereas the blue coloured circle shows the phases of traditional software development. The main difference is that some of the phases in the Low-

Code/No-Code approach can be completed faster in comparison to the traditional agile approach (al Alamin et al., 2021). The market researcher Forrester claims, that Low-code platforms accelerate software development by a factor of five to ten compared to conventional methods (Rymer, 2018). Trigo et al. (2022) conclude that there is a time saving, but not to the extent claimed by Forrester. The authors assume a three times higher development speed.

2.4 Founders perspective: Drivers and inhibitors for utilisation of No-Code solutions

2.4.1 Drivers and beneficial reasons for utilisation of No-Code solutions

Kass et al. provide a detailed table about drivers and inhibitors for No-Code solutions, based on an intense literature review. This table is shown in the Appendix and will be referred to in the discussion later. Kass et al. (2022) elaborated the drivers and inhibitors for the adoption of No-Code in detail, along the Diffusion of Innovation Framework. The factors most cited in their literature review, regarding the adoption of No-Code platforms, are *"improved performance metrics of the software development process"* (Kass et al., 2022, p. 199), followed by *"reduction of entry barriers for application development"* (Kass et al., 2022, p. 199) and third, most frequent factor, is "*ease to develop applications"* (Kass et al., 2022, p. 199). Summarised, the main driver belongs to the category relative advantage, as described in Table 1. These findings are supported by a research of Brühl et al. (2023). 120 startup companies have been surveyed based on a questionnaire where they assessed the usage of No-Code tools and platforms. The preeminent benefits provided by the aforementioned authors are less development time with an approval of 73,3%, followed by lower development costs with an approval of 57,5% and third most common is simplicity of use with an approval of 54,2%. However, out of the 120 participating startups only a small fraction of 12% indicate that they believe to have no advantages using No-Code Tools (Brühl et al., 2023).

Lu (2020) observes that founders decide against a developer team for time- and especially cost reasons and use No-Code solutions for startup founding. It can be observed that more and more companies are also founded by non-technical founders. Their previous deficit of technical knowledge is compensated by No-Code tools. Rafiq et al. (2022) studied the adoption of No-Code in software startups. The case study reveals that No-Code is primarily utilised for prototyping, experimentation with novel features, the development of internal solutions or to generate a by-product. Noteworthy, the authors observed an unexpected finding: No-Code is not utilised for the main product, but only for by-products or experimentation. The explanation

given is that No-Code providers are unable to cover the specific requirements, since they are quite unique.

According to Sufi (2023) the benefits for using No-Code platforms are that they are e.g., easy to learn, address developer scarcity, enable to test ideas in a cost-efficient way, increase development speed, provide proven design, show best practices, and enable the integration of external sources. Summarized the core benefits of Low-Code/No-Code platforms to solve problems are that 1) AI powered tools can be used to enhance these Low-Code/No-Code platforms and thus automate tasks, 2) enhanced user experience is provided through intuitive, simplified, drag and drop features, 3) the ability to integrate external tools and sources such as databases or cloud platforms, to create applications with higher complexity. Furthermore, 4) the adoption of this technology increases because of its easy usage and ability to solve individual business need fast.

2.4.2 Inhibitors and reasons for rejecting No-Code solutions

The main inhibitors to the adoption of No-Code platforms identified by Kass et al. (2022) include limited customization and flexibility, absence of scalability, challenges in ensuring compatibility. Summarised, the main inhibitors identified in the literature review by Kass et al. (2022) belong to the category compatibility and complexity, as mentioned in Table 1.

One startup CEO participating in the study by Rafiq et al. (2022) cites his experience that investors are not willing to invest if the product was created solely by No-Code. He sees lack of knowledge and understanding about No-Code solutions capabilities as cause.

The main reasons for not using No-Code tools, in the below mentioned study conducted in Slovenia, include doubts about feasibility and lack of knowledge about the development of No-Code tools and understanding the platform and associated platforms. In addition, there are concerns about becoming dependent on one No-Code vendor (Beranic et al., 2020). The No-Code platform Bubble for example introduced its new pricing structure in May 2023. Customers can either convert to the new price structure or maintain their current plans for up to 18 months with a 10% increase (Bubble, 2023).

The preeminent disadvantage according to the survey by Brühl et al. (2023) is limited customization, followed by limited functionality, which was indicated by more than 50% in each case. The third most common is vendor lock-in. About one third of the participants stated General Data Protection Regulation (GDPR), due to hosting outside the EU and approximately

28% the lack of ownership of the source code as disadvantage. A small fraction of approximately 9% indicated to believe that using No-Code has no disadvantages (Brühl et al., 2023).

According to Sufi (2023) the limitations for rejecting No-Code platforms are that they e.g., 1) create a shadow IT, since everyone is able to build applications without the need to consult the IT department, 2) become dependent on a particular platform, also known under the expression vendor lock-in, 3) are inappropriate for mission critical processes, 4) missing flexibility, since the available development possibilities are limited, 5) missing On-Premise options, since No-Code solutions are cloud based and maintained. Therefore, they are unsuitable for the use of private or secret data for instance such as military or governmental context.

Once vendor lock-in and the resulting cost commitment is appropriately addressed, a broad adoption for Low-Code/No-Code platforms will be reached (Sufi, 2023).

2.5 Investors perspective: Beneficial and restricting factors for securing an investment

There are different types of investors, being engaged in different startup stages, as illustrated in Figure 7. Therefore, their goals and factors for deciding for an investment may differ (Alemany & Andreoli, 2018).

2.5.1 Types of investors along the startup phases

The type of investor changes over time through the different startup stages. An explanation of the four illustrated phases can be found in chapter 2.1.1. In the initial phase, the entrepreneur is often dependent on himself, by making use of own financial resources or relying on close people such as friends or family members. In this context also referred as the three F's: family members, friends, or fools. An incubator can help to accelerate the development process in the beginning and provide the opportunity to network. In the startup phase the company is a bit more developed and generates revenue. Therefore, the list of potential investors expands, including for instance angel investors, accelerators, and venture capital firms (Alemany & Andreoli, 2018). In the growth and maturity phases, the list of potential investors expands even further as shown in Figure 7. Since this work focuses primarily on the early phases, the later stage investors will not be further elaborated at this point.

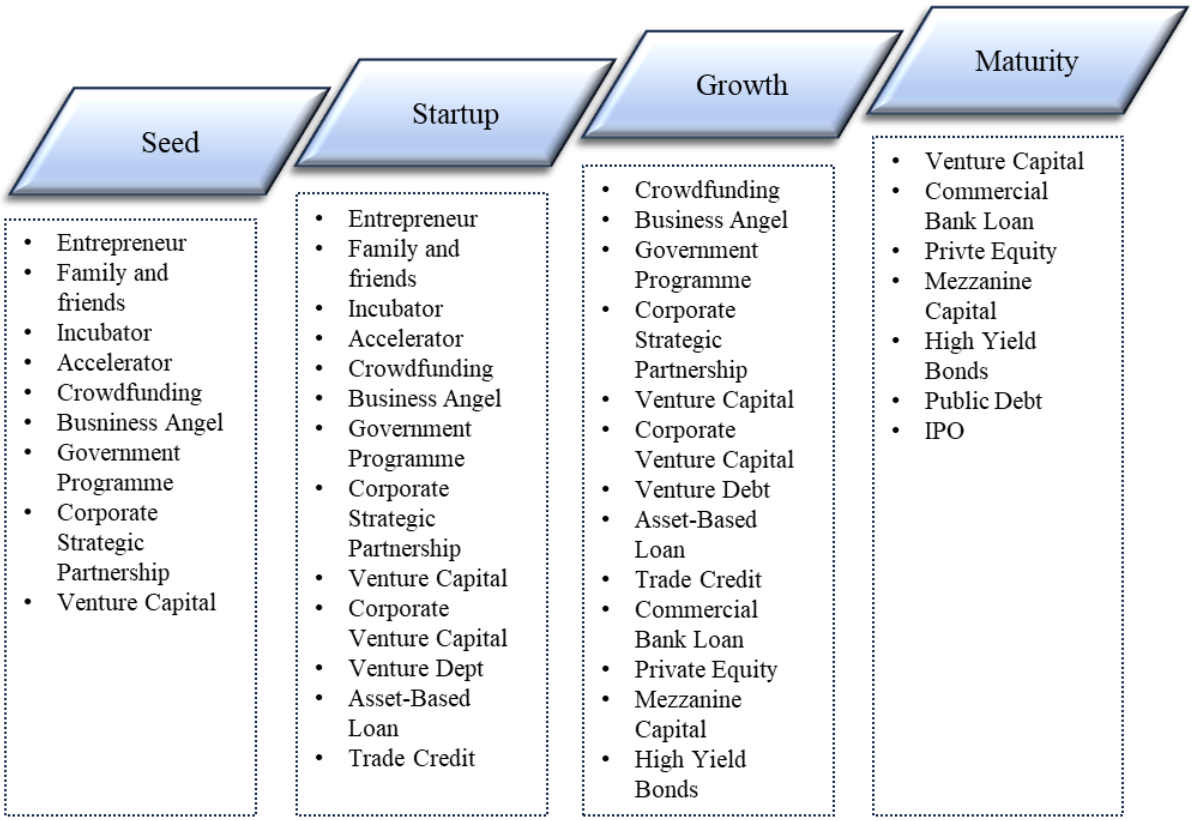

Figure 7: Sources of funding by venture development phase
Source: Adapted from Alemany & Andreoli *(2018, p. 17)*

2.5.2 Criteria investors consider when making investment decisions

Deal screening criteria used for example by Venture Capital investors can vary on various factors like preferred stage, size, region, portfolio related, or deal structure related (Petty & Gruber, 2011). Tyebjee & Bruno (1984) identified four distinct areas inside the business plan that are employed to assess the level of risk and the possible financial gain. The four components encompass the capability to effectively manage marketing, competitive advantage and distinctiveness of the product, the management team and its skill balance and external risk elements that are outside of control. The authors provide technology obsolescence as an example for the latter.

Carpentier & Suret (2015) analysed 636 startup proposals and concluded that angel investors most common reasons for rejection are associated to market or execution risks. Furthermore, rejection criteria for founders with a lower level of experience are mostly associated with product or market reasons. Petty and Gruber (2011) categorized the reasons for rejection in product, market, financial, team, investor specific reason and other. According to their categorisation the section product includes complexity as a subcategory and the section team includes the reasons inexperience and incomplete management as a subcategory. Vogel et al.

(2014) examined the diversity of entrepreneurial teams and its implications on investments. The authors subdivide the dimension diversity into task-oriented diversity and relations-oriented diversity. Blank & Carmeli (2020) investigated the influence, the team composition of the core team has on external investment. Therefore, they assessed the experience of the core team using the variables entrepreneurial experience, industry specific experience and managerial experience. The authors found out that industry specific experience in particular has positive impact on the ability to secure external investment.

2.6 Prognosis of future demand for software developers

Nevertheless, there are contradictory opinions on the future development of coding. The CEO of Github proclaims *"the future of coding is no coding at all"* (Peterson, 2017). Whereas the CTO of Coveros has an alternative viewpoint and proclaims that

> *Low-code is not the future of code. It certainly has a place in the future and will be leveraged to make many applications. It will not replace other ways of creating software because low code breaks down when the solution's complexity increases. We saw the same thing with Visual Basic in the '90s. VB was valuable, and a lot of software was written in VB. In the end, it was the complexity required by some applications that caused VB to break down and no longer be a good solution. Low code will be the same* (Brocoders, 2021).

Hughes (2023) examines the impact on developers and points out the smartphone development as an example. He points out that smartphones have not replaced the demand for computers and the technology development and availability of tools such as ChatGPT or AI tools will not make developers unemployed, but rather increase the demand for developers. Furthermore, specialists are needed to further develop these tools for non-tech-savvy people. This assumption is supported by a forecast from Evans Data (2022). The global number of developers is estimated to be 27.7 million in 2023 and will grow by 3.6 percent to about 28.7 million by 2024. The rate in technological development of No-Code solutions, particularly when integrated with AI or ML technologies, underscore the necessity of IT experts who possess a comprehensive understanding of risks and limitations (Hurlburt, 2021).

3 Methodology of Qualitative Research with expert interviews

This thesis applies selected methods of qualitative social research to gain empirical insights that is used to answer the research question in chapter 1.3.2 and thus gain a deeper understanding of the research subject. Simultaneously, this allows to place the empirical findings in the theoretical foundations from chapter 2, as suggested by (Ryan et al., 2007). In the subsequent chapters, the chosen research design, the aim of the applied methods and further methodological approaches will be discussed.

3.1 Research design

A research design is a methodical strategy for achieving the research objective. It encompasses the research objective with all its steps to be executed within the research process, e.g., the generation of data or the analysis. By guaranteeing that the research question is specifically addressed and answered within the parameters of the task, the research design thus goes beyond a work plan (Yin, 2018). It concerns e.g., what questions are to be investigated, decisions regarding pertinent data, data collection methods, and analysis of the findings (Philliber et al.,1980), as cited in Yin (2018, p. 66). More specific, a research design must include the following "ingredients" (Schwarzer, 2001) in order to meet the five standards for scientific quality: 1) a specific research question, 2) details on the topic's scientific and/or practical importance, 3) the current state of the field, 4) the theoretical underpinnings, 5) and the methodology (Schwarzer, 2001). The aforementioned ingredients one, two, three and four are described in detail in chapter 1.3.2 and chapter 2 and thus shall be not repeated here.

Research Method

Researchers have a variety of data collection strategies at their disposal when conducting qualitative studies. These data collection options include i.e., questionnaires using open-ended questions, written text material such as e-mails or unstructured and semi-structured interviews (Ryan et al., 2007).

The methodological objective of this thesis is to apply the data collection method of expert interviews, followed by the analysis method of qualitative content analysis, which is conducted according to Mayring's principles (Gläser & Laudel, 2009; Mayring, 2015). In this thesis, an

> *Expert' describes the specific role of the interview partner as a source of specialised knowledge about the social issues to be researched. Expert interviews are a method of unlocking this knowledge* (Gläser & Laudel, 2009, p. 12).

Since special and practical knowledge in the field of No-Code is needed in this thesis, an expert interview, as one of several guided interviews, is the adequate choice according to Döring & Bortz (2016).

Within the analysis phase in qualitative research, various methods can be chosen, such as grounded theory methodology, objective hermeneutics, or qualitative content analysis (Döring & Bortz, 2016). The analysis method of qualitative content analysis has been chosen since it is suitable for processing large amounts of material that can be interpreted qualitatively and thus classified in terms of its meaning. The analysed text material is first generated by collecting data e.g., through documents, open interviews, or newspaper articles, and then subject to a strictly rule-based and thus verifiable analysis (Mayring & Fenzl, 2014).

In line with Kaiser (2021), who provides a ten-step-approach to plan, conduct and analyse expert interviews adequately, first, an 1) interview guide was established, 2) interview questions were pre-tested, 3) interviewees selected and contacted, 4) expert interviews were held, and 5) recorded. Then, 6), the interview results had to be secured. This was done by transcribing the recorded interview. In step 7), the transcribed text had to be coded, in order to 8) identify the experts' core statements. In step 9, the data basis was expansioned and finally, the author conducted an interpretation and generalization based on theory (Kaiser, 2021). Subsequently, step one shall be explained in more detail.

3.2 Conception of the interview guideline and its pre-testing

For the creation of a "semi-structured interview" (Döring & Bortz, 2016) guideline, initially exclusively theory-oriented categories were formed. This deductive, theory-oriented approach, also called a-priori-category-construction, requires extensive prior knowledge (Kuckartz, 2018) gained by the author through an in-depth literature review presented in chapter 2. This was done by following the steps, described in (Mayring, 2015), inclusive e.g., the definition of codes/categories, anchor examples, coding rules, interpretation and resulted in the deductively generated category system. Based on this procedure, the interview guideline was set up. The categories and subcategories obtained are presented below and show the final decision after having conducted a pre-testing.

Category	Subcategory
Experience & Suitability	Drivers for usage
	Inhibitors for usage
No-Code usage according to Startups Stages	No-Code as adequate approach
	Traditional coding as adequate approach
Skillset	Redundant competencies
	Consistent competencies
	New competencies
Investment Decision	Factors fostering investment
	Factors inhibiting investment

Table 2: Interview guideline categories
Source: Own illustration

By pre-testing of the interview guideline with one selected expert, the author ex-ante intended to optimize the instrument of collecting data (Weichbold, 2014).

The advantage of a semi-structured interview is that the set-up interview guideline with its open-ended catalogue questions, provides the interviewer with questions and their sequence, but also allows him or her, for example, to slightly modify the questions, leave some of them out, discuss them in more detail, depending on the interview situation (Döring & Bortz, 2016).

3.3 Interview Sampling

There are various sampling approaches, e.g., quota sampling, sampling plan, convenience sampling, or the self-selection approach, where the researcher gains access to experts via an Access-Panel-provider. Here, the qualitative sampling plan-approach was highly adequate for getting an in-depth knowledge from experts within a particular industry and thus chosen. As usual in the sampling plan-approach, the author deliberately decided for a particular composition of the candidate sample on the basis of research-specific relevant characteristics such as profession or age (Döring & Bortz, 2016). In contrast to quota sampling, in which the target population is replicated quantitatively in the sampling composition, the selection of cases or candidates following the qualitative sampling plan, should be based on the information content, the cases or candidates are able to provide (Döring & Bortz, 2016).

Definition of expert: Candidates were defined as experts and thus selected, when they fulfilled the criteria of having knowledge of the No-Code industry due to their profession and or belonged to either the group of investors or founders (Döring & Bortz, 2016). In the interview itself, a previously prepared interview guide (Döring & Bortz, 2016) was used as shown in chapter 3.2. After having selected the sampling method, the way to gain access to selected experts and the manner in which they were contacted, is shown below.

Procedure to gain access to experts and the way of contacting them

In the first step, an analysis was conducted on LinkedIn to identify relevant experts. During the research on LinkedIn, several communities on the topic of No-Code were discovered. In addition, there are websites such as WeLoveNoCode, which explicitly provide No-Code experts on a freelance basis. Other websites like No-Code Founders and No-Code Alliance list startups that use No-Code tools to build their solution or product. Using the information from these websites, a consolidated Excel list was created that included company names and a listing of their founders. Likewise, employees and founders of No-Code platforms were identified through LinkedIn and included in the list. In addition, the owners of YouTube channels that explicitly deal with the topic of No-Code were added. In the next step, missing information was added through LinkedIn research. At the same time, a list of angel investors, incubators, accelerators and early-stage venture capital firms with a focus on tech investments was created. In the last step, 117 people from both lists were contacted via LinkedIn. Of the 117 people contacted, 18 responded and 12 were willing to be interviewed. One interview was held with two people at the same time. Therefore, participants with a computer science background, participants without a technical background as well as participants with in-depth knowledge of startup investing were interviewed.

Interview	Participants Profession / Role	Educational Background
A	CEO of Software Development Agency	Marketing
B	CEO and Co-Founder of several software companies & Investor	Computer Science
C	Founder of a Webdesign Agency	Philosophy
D	Software Engineer	Computer Science
E	Chief Technical Officer; No-Code Instructor	Computer Science
F	Chief Strategy Officer & Chief Operating Officer of a No-Code vendor	Business Administration
G	Angel Investor	Business Administration
H	Software Developer & Founder of a No-Code Agency, No-Code Instructor	Media Studies
I	Founder of No-Code Agency	Management
J	Partner early-stage Venture Capital firm	Business Administration
K & L	Director of Startup Accelerator & Head of Acceleration	Business Administration & Economics

Table 3: Overview of interview experts
Source: Own illustration

3.4 Data collection method: expert interviews

Demonstration of the interview situation

Due to the physical distance between the interviewer and the interviewees, the interviews were conducted via online-interview (Döring & Bortz, 2016) using the video conferencing software WebEx, Microsoft Teams and Zoom. In line with Döring & Bortz (2016), all interviews were expert interviews, specifically individual interviews in attendance of the author as interviewer and the selected expert. The conversations, which all were spoken in English, were audio recorded, and the transcription function was enabled (Döring & Bortz, 2016). Following that, the interviews were manually post-processed and reviewed for errors.

Data Preparation: Rule-based Transcription

The original, first transcription of each interview was automatically generated by the chosen online conference software, enabling their transcription functionality. In a second step, the author listened to the audio files and modified the automatically transcribed text manually. The transcription was conducted rule-based and was therefore completely in line with the 14 rules from Kuckartz (2018), shown in his figure 27. Due to the need of anonymisation, each expert is shown with a unique letter from A to L, depending on the sequence of when the interview was conducted.

3.5 Qualitative data analysis procedure

To conduct the analysis of the collected data, the author followed the recommendations of Döring & Bortz (2016), Kuckartz (2018), Mayring (2015). The procedure itself can be carried out either by

1) a 'manual qualitative data analysis' (Döring & Bortz, 2016, p. 607),

2) using office programs such as Microsoft Excel or Word – known as 'qualitative data analysis with general-purpose software' (Döring & Bortz, 2016, p. 608) or

3) utilizing a 'computer aided/assisted qualitative data analysis software' (Döring & Bortz, 2016, p. 608) such as MaxQDA, Atlas.ti or NVivo (Döring & Bortz, 2016, p. 608-609).

In the present case, MaxQDA was used initially for the data management and the set-up of the category- and code-system. Afterwards, the processed data was exported in Excel and finally analysed.

Step-by-step approach for the qualitative content analysis

Within the phase of analysis, codes are generated, and similar ones allocated to superior categories. The aim of this procedure is to reach a higher abstractive level, which the original raw data material cannot provide (Döring & Bortz, 2016).

This requires the understanding of relevant terms, the author utilized, following Kuckartz (2018). A category or code can be defined as *"the result of the classification of units"* (Kuckartz, 2018, p. 31). Differently said, when establishing categories, a clear definition of what kind of text content fits in a category is needed (Mayring, 2015). Furthermore, a coding frame or category system is understood as the totality of the defined categories (Kuckartz, 2018). Coding is the allocation of text passages to a corresponding category/code (Döring & Bortz, 2016; Kuckartz, 2018).

The required codes and category system were first theory-based generated, in other words, a deductive approach applied, as demonstrated in chapter 3.2, and secondly completed and modified with an inductive approach out of the interview transcripts (Döring & Bortz, 2016). The inductive approach generates codes out of the collected interview data, while the deductive approach is theory based and generates them out of the literature. Both ways follow the same recipe and differ, starting with another material (Döring & Bortz, 2016). The deductive approach is already displayed in chapter 3.2, when creating the interview guideline and thus, will not be further explained in more detail at this stage.

Inductive code generation

Before beginning with the inductive code generation, all transcribed texts were studied thoroughly. Then the steps suggested in chapter 5.5.2 in Mayring (2015) and recommendations from Döring & Bortz (2016) were applied. The author marked specific passages in the text, which seemed to be relevant for answering the research question. In qualitative research, these text passages are termed units of analysis. Each unit of analysis shall be well-defined and fulfil relevant criteria. Thus, relevant text passages were allocated with a category (Döring & Bortz, 2016), the text once more reviewed and where required, new categories were built. The inductive category system has been finalized by paraphrasing coded interview statements, deleting some text passages with less relevant content, then text passages were generalized and reduced/subsumed to obtain a higher level of abstraction. Similar or identical paraphrases were bundled, which resulted in the final inductive category system. This is completely in line with Mayring (2015).

Having generated these inductively categories and subcategories, as allowed in qualitative content analysis, the deductively developed category system was completed, adjusted (Mayring, 2015) and thus, the final category system built. Only after having the final category system, the whole data material was coded and analysed, interpreted, using MaxQDA and Microsoft Excel, as described above. This resulted in the findings described in the subsequent chapter.

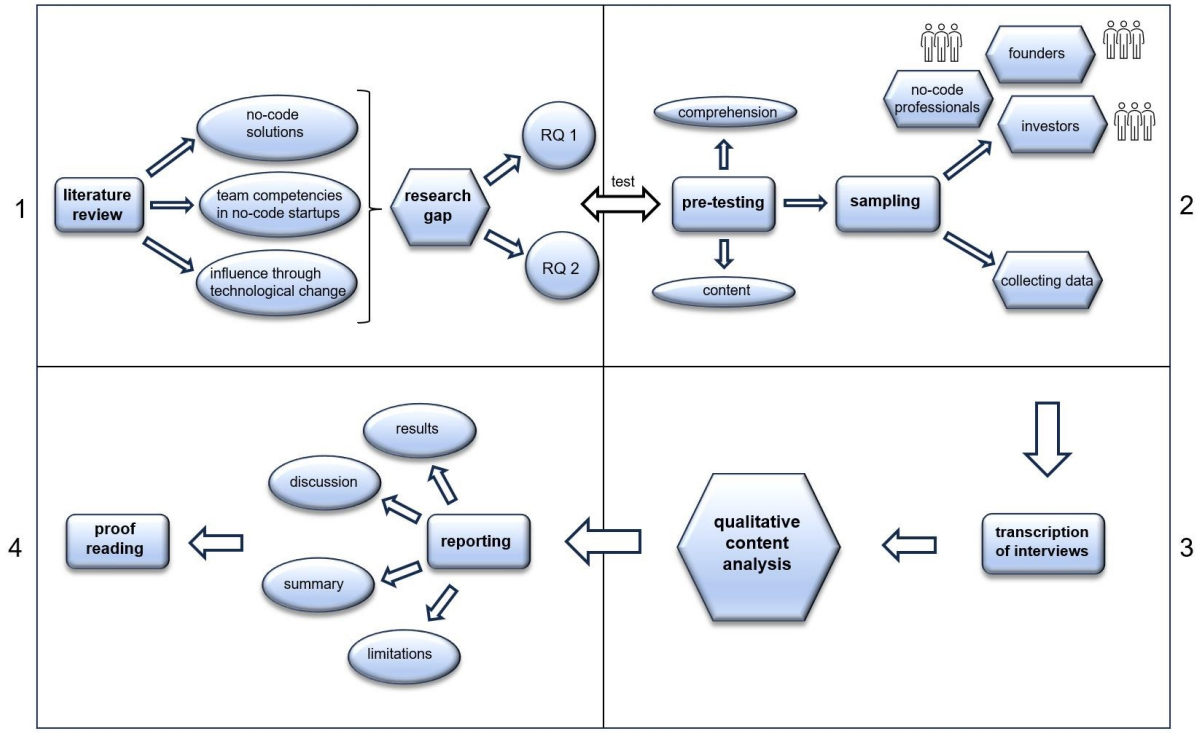

Figure 8: Methodological procedure
Source: Own illustration

4 Results from expert interviews

The section provides important findings out of expert interviews with founders, No-Code providers, and investors. It presents the motivations behind the utilisation of No-Code solutions as well as an examination of existing inhibitors that impede its adoption. Furthermore, the current software development approach in startup companies is presented. In addition, this analysis examines the impact of No-Code on the future workforce and the shift of essential competencies within startup teams. First, relevant findings for RQ1 are demonstrated.

Category	Subcategory
Factors influencing the acceptance	Drivers for No-Code utilisation
	Inhibitors for No-Code utilisation
State-of-the-Art software development approach across the stages of startup development	No-Code as adequate approach
	Traditional coding as adequate approach
Future Development of No-Code technology	No-Code and AI
Impact of No-Code Movement on Workforce	Reduction of workforce
	Shift of essential competencies within of a core team

Table 4: Overview categories from expert interviews
Source: Own illustration

4.1 Factors influencing the acceptance of No-Code solutions

There are various positive and negative factors to be found influencing the utilisation of No-Code solutions on the one hand and the investors' acceptance and thus their willingness to invest in startups using No-Code technology on the other hand. First, positive utilization factors are demonstrated.

4.1.1 Drivers for No-Code utilisation

After having applied the methodological steps in chapter 3 on the responses obtained by experts, three categories were identified that can be seen as drivers. These three categories are enablement, enhanced operational effectiveness and simplicity.

Enablement

Many experts underline the importance of the ability to build software without requiring extensive knowledge or coming from a technical background, such as participant F. He represents the point of view, that:

> *[…] anyone can learn anything and you're just a little more efficient until you get there […]. But to be actually more realistic, I would say the other benefit is a democratization of certain skills and certain capabilities and No-Code allowing you to reap the benefits without having the skills that usually only experts and specialists would have* (Interview F, Line 30-33).

Participant K supports this view by stating "*[t]hese solutions are unbelievably powerful, I mean, it's amazing. You can have a degree in management, have absolutely no idea of how-to code something and you can make your website"* (Interview K, line 136-139). In the past decade, entrepreneurs who lacked technical skills faced the dilemma of developer scarcity on the labour market. In addition, they were not capable of estimating the developer's actual expertise. No-Code might address the dilemma of developer scarcity since it diminishes the requirement for an in-house developer (Interview G, line 41-46). Participant D also addresses the fact that in-depth experience in the core team is not as important as it used to be, mentioning:

> *You needed a core team, that knows this well and they had to be of a certain seniority, but with No-Code you have an advantage. Now you can do this with a less experienced team, and that is where it affects the skills at most* (Interview D, line 365-367).

No-Code creates possibilities that were previously impossible and opens an entirely new market for developing applications fast (Interview C, 43-47, 50-52; Interview A, 147). This advancement contributes to the fact that a junior developer or anyone with knowledge about the usage of a particular No-Code solution can perform tasks. As a result, costs for expensive senior software engineers can be saved (Interview D, 372-375). Lower development costs result in less initial capital required to validate a potential business idea. It postpones the need for investment and lowers the barrier for launching a business and revenue can be generated immediately (Interview H, 63-65; Interview I, 131-134 & 212-213). Participant F goes even further and does not limit the user group to non-technical individuals. There are also founding teams that have an exclusively technical background. No-Code solutions can assist them in grasping the commercial dimensions. It provides a basis for the development of expertise on both ends, for technical and non-technical individuals (Interview F, line 204-208). This is due to the fact, that these tools provide logic as a service (Interview E, line 94-96).

Enhanced operational effectiveness

Many interviewees stated speed as an important factor, resulting in a shortened time-to-market as crucial advocates of No-Code. The increased speed allows higher flexibility as well as more and faster iterations. Thus, more is accomplished in less time, which in turn can provide a competitive advantage and substantial cost savings (B, 34-35; C, 105; D, 507-511; G, 124-126; H, 13-17; I 151-155). In particular for early-stage ventures, time-to-market is critical. Once an idea is validated, the competitors attempt to enter the market as well (A, 38). Interviewee D provides an example in which a task with coding usually requires three to five days and now can be accomplished in half-day using No-Code (Interview D, 78-79). Interviewee E uses Lego bricks as an illustration. The utilisation of existing Lego bricks reduces the need to build everything from scratch, which benefits everyone. However, if there is one brick missing, it is possible to code it and incorporate it with plug-ins, to minimise the waste of time. Reusable components can and should be utilised (Interview E, 81-83). For expert F, entrepreneurs using No-Code solutions can be seen as problem solvers, since they have a clear goal and show a *"can-do-attitude*" (F, 300-304), using possibilities in a hands-on manner to get their problem solved. Concluding, the factors speed, innovation and problem-solving can be seen important to entrepreneurs, using this technology.

Simplicity

Any kind of fundamental marketplace, website, or software application (H, 23-24), or if it is sufficient to implement simple if-then rules or alerts and notifications (D, 208-212) No-Code solutions are particularly suitable for this purpose (D, 211-212). In a nutshell, everything that includes automation especially in small businesses (E, 105). E states that it is a fallacy to believe that entrepreneurs always must invent something entirely novel. Sometimes it is sufficient to combine something preexisting in a novel setup to create value. That is exactly where No-Code can add value (E, 53-57).

4.1.2 Inhibitors for No-Code utilisation

On the responses obtained by the experts, four categories were identified that can be seen as inhibitors. These four categories are complexity, insufficient maturity level, security and lack of experience.

Complexity

As complexity increases, suitability for No-Code solutions decreases. A mentions in this regard computation and user interface as well as the number of users as factors, which might slow down performance (A, 60-64). H adds high volume base applications and G believes that highly specific or highly innovative applications are not particular suitable (G, 70). If someone wants to create a unique product No-Code is the wrong approach as it mostly offers just a pre-built kit that allows to develop existing things in a novel way (E, 53-57).

Insufficient maturity level and vendor dependency

D, E and J claim that the current No-Code development is not sufficiently advanced (D, 383; E, 34-35; J, 81-82). Therefore, K is sceptical about outsourcing essential core elements or critical processes. Once a sufficient level is reached, partial outsourcing is conceivable (K, 30-32). Whenever there is a critical process, e.g., a money-making process, it should be written in custom code since it is easier to ensure a higher level of traceability and it is simpler to set up a test environment, with the advantage that errors can be detected more quickly. The size of the company is curial in this case. D recommends for small ventures to continue utilising No-Code even for critical processes, for the simple reason that cost-performance ratio is still out of proportion (D, 191-196). F argues against this view as he considers it outdated and asserting, that No-Code vendors have made significant advancements in order to become more reliant (F, 72-76). Another issue is that some providers do now allow to export the generated code, which can be seen as a deterrent at first. The source code belongs to the No-Code tool, nevertheless, the built solution belongs to the creator (I, 228-231). C, however, claims that software applications must always be reworked and rebuilt. Therefore, he considers it as a less severe issue (C, 83-85). Whereas B does not consider the current maturity level as an inhibitor for utilisation and investment. Dependency on the vendor is more of an issue. From an investor's point of view, the decisive factors are in the area of business risk and legal (B, 174-177; J, 192-193). For instance, the missing ownership of Intellectual Property (IP) that can have greater negative influence. B illustrates this with two possible scenarios. Scenario one: the ventures success is dependent on the No-Code provider's success. If the No-Code vendor discontinues his service, the fundamental basis is taken away. Second scenario: unpredictable price swings for their licenses can have significant impact on the financial viability to generate profit or even make it impossible to operate sustainable (B, 165-176).

Security Aspects

The utilisation in regulated industries should be carefully considered. The experts advise against examples such as dealing with financial data or healthcare information, due to regulatory requirements such as data hosting (C, 131-132; E 88-91; I, 67-68). F adds security, military-industrial complex, or areas of public sector (82-85) and summarises data protection and identity and data security as area where people must be cautious. Nevertheless, F considers it possible to use No-Code in these areas, but it is not the most suitable. The benefit of No-Code lies in speed rather than reliability.

Lack of experience and trust

Currently, No-Code is perceived more as a liability when it comes to an investment (A, 217; K, 46-51). However, this is primarily due to a lack of experience. A mentions that many people with a computer science background are sceptical about No-Code. One reason for this is the fact, that some No-Code vendors present themselves as competitors to software developers, resulting in a decrease in acceptance (A, 223-226). D confirms this observation with a personal experience. There are instances where software developers chose to leave the company due to the fact that they refused to work with No-Code (D, 326-327). Moreover, there are still too few success stories. Once there are several successful ventures on the market, shift in mindset is likely to occur. The hypothesis of C about missing success stories is that investors still have a negative perception and therefore are still cautious and reluctant to invest in a venture that outsources core technology by relying on an external provider (C, 253-257). C claims that there are already success stories, and a rethink is taking place. As an illustration, he cites the online marketplace Acquire.com, that facilitates the sale and acquisition of tech startups (C, 281-287).

4.2 State-of-the-Art software development approach across the stages of startup development

All interviewees agree that relevance for No-Code tools is highest in the early stages. L stresses the argument, that *"[d]uring the pre-acceleration idea stage, No-Code is a must"* (L, 83), being supported in this point of view by interviewee K (135-136). If founders do not use No-Code solutions at the beginning, Interviewee B demands a justification from founders why they refrain from using No-Code solutions. In the first step, his assumption is that founders do not understand No-Code technology and the associated benefits such as speed and low development costs. Without an appropriate justification, it is an exclusion criterion for an investment (B, 198-

202). The director of a startup accelerator, K, states that No-Code solutions offer at least the possibility in the initial phase to test and validate potential business ideas. Otherwise, it serves as an indicator for insufficient entrepreneurial spirit. It allows to draw conclusions about founders. F associates the willingness to use No-Code with a can-do- and hands-on mentality as well as problem-solving competency. It indicates the ability to solve problems fast and in an innovative way (F, 300-304). In some areas No-Code tools are already commoditized e.g., website development (K, 136-139). According to participant G, not using No-Code for any kind of generic commerce is an exclusion factor of investment, since this is an indicator that resources are not allocated efficiently (G, 185-189).

However, Interviewees A, B, C, D, F, G, H, I, J, and K believe that relevance is present in all stages. A cites FINN, a Munich-based German mobility startup as an example of a company that utilises No-Code solutions in all stages. Nevertheless, the interviewees perceive that the highest relevance is in the early stages, when founders are still looking for product-market fit, i.e., seeking whether there is a market. Meaning if their solution or product solves a customers' problem. According to D it is important to adapt quickly at this stage and No-Code enables it. F recommends the use of No-Code especially for the creation of a MVP (F, 123-124). In later stages, the use of No-Code tools becomes more specialised and specific and therefore decreases a bit (F, 132-134; G 113-114). F and H also believes that the usage shifts from being the core element of value creation towards serving as a support of value creation (F, 148-149, H, 45-47). According to E's perspective, No-Code tools are only relevant in the initial phase, followed by a mandatory transition to code (E, 157-158 & 164-165). He explicitly emphasizes, that he is referring to the initial product phase with his statement, not the startup in general, since one startup is able to offer multiple products or solutions to their customers. However, they are only suitable in the initial phase of product development. Once there is traction, meaning customers are willing to pay for the product, then E suggests the transition to code (E, 157-162). He recommends moving from No-Code to Low-Code and then to code, that is how it will evolve in the near future.

There are different views on the need and appropriate time to transition to code. H suggests that, once experiencing system-related limitations, it is advisable to transition to code, since it can be more time-consuming to build a workaround compared to transitioning to custom code or when the stage is reached, that custom code is faster. At the latest, when a startup reaches the maturity stage priorities shift, speed is no longer the focus. Optimising the validated product in order to achieve the best possible version is the new priority (H, 36-40). D on the other hand

suggest a switch once product-market fit is achieved (D, 92-93). The usage decreases in Series A, at the latest Series B as systemic possibilities have been exhausted and critical money-making processes should not be dependent on external providers (D, 451-459). I and K make it application dependent, respectively dependent on the use case. In the case of a small venture a change to code is not necessarily required. However, if the goal is to scale and become the next Facebook for instance, transitioning to code is inevitable. Outsourcing such a fundamental component of a business is not realistic (I, 82-87; K, 13-16).

Technological insignificance

Also, it shall not be missed out, that the underlying technology is usually not immediately apparent what is running in code and what is not (A, 114-116; D, 304-305). Furthermore, the customer does not care about the underlying technology, but rather is interested in the solution to his issue.

4.3 Future development of No-Code movement and AI

B and E indicate that they observe the occurrence of two trends happening: one trend is No-Code, the other is AI (B, 275-276; E, 128-131). No-Code eliminates the need for manually writing code since everything is developed visually. Arising developments such as GPT and LLMs belong to the AI trend. It is worth noting, that those new developments include the ability to create code without human intervention. This raises the question why someone should opt for No-Code platforms if there are solutions on the market that can automatically generate code (B, 223-232). E even argues that *"(...) it's not called No-Code, now it's called AI"* (E, 8-9), and indicates that a merge of the two movements is taking place. I assumes that No-Code vendors include assistants such as the co-pilot technology or similar tools, which will enhance speed even more. The degree of integration depends on the associated costs, as integration remains expensive at the moment. Nevertheless, the providers are currently in a testing phase. According to his estimation it enhances developers' productivity by 50% and therefore has significant impact (I, 321-329). The participating COO of a No-Code provider claims that since they introduced generative AI it only takes 18-60 seconds to build an app with their No-Code solution, which confirms that a merge of No-Code and AI is taking place (F, 168-171), supporting E and I. According to his estimation, there will no longer be a distinction between the two trends, in about three to five years. Interviewee F states that *"[...] one perspective would be generative AI will kill No-Code, but I would prefer to say they will become something*

else. So, generative AI will be dominant, but we'll carry on the No-Code idea and legacy […]" (F, 187-189). The advantage is not only speed, rather a greater number of applications that are feasible to be developed, including applications that might not have been feasible before (F, 171-176). It will still be software development, just with a larger variety of Tech Stacks (A, 226-230).

4.4 Impact of No-Code solutions workforce and skillset

4.4.1 Reduction of workforce

An increase in the digitalization of products and processes is globally taking place. Yet the combination of those two trends is responsible that the number of the software development workforce declines in the future. No-Code alone is not able to do that (E, 128-131). B already observes a significant reduction of development costs, down to ten percent in contrast to eight or nine years ago (B, 276-279).

Due to the increase in efficiency, Interviewee B, holder of a degree in computer science, postulates that in the future only one developer is needed for the work that required a team of ten developers (B, 256-258). Consequently, demand for this skillset will decline in the long term. He hypothesises a reduction by a factor of ten (B, 264-265). C follows this direction of argumentation expecting a decrease and points out that there is a lack of awareness regarding this matter: "*I think it should scare software developers more than it does, because it replaces a lot of the work that they get paid enormous sums of money"* (C, 197-198). He expects a profound shift in the business landscape over the next ten years that will impact all types of businesses and will result in software developers being less in demand (C, 198-200). In addition, not only software developers are affected, certain positions in the area of legal or marketing, explicitly e-mail marketing, are also affected by this development (C, 210-212).

C and E hold opposing opinions towards which skill level of developers will be in demand in the future. C claims that the demand for mid-level developers will decrease due to new technologies. He mentions No-Code tools like ChatGPT or LLMs which are able to perform the tasks and even produce better results than humans do. This development impacts mid-level professions the most in his view (C, 209-217). Interviewee E on the other hand assumes that mid-level developers are in high demand, since they have sufficient experience, but notably lower cost compared to senior developers. Employing one senior expert with extensive

knowledge to make the elementary decisions will be sufficient. For this reason, he emphasizes to recruit an experienced CTO early (150-153).

4.4.2 Shift of essential competencies within a core team

Three categories concerning the changes coming from No-Code and influence the skillset in small teams of early-stage ventures can be found. These are: competencies declining in significance or required at a later stage, competencies that continue to be in demand, competencies that are gaining in importance.

Competencies with declining in significance or required at later stage

So far, the prevailing opinion is, that technical skills are a prerequisite for undertaking any type of business (G, 144-145). This prevailing viewpoint has changed over the last years since the importance of programming ability is no longer as critical, due to the fact that it is possible to accomplish a broad spectrum of tasks with No-Code solutions (B, 146-147). Therefore, the evaluation of a team's calibre is no longer predominantly determined by technical ability, e.g., having a highly experienced software developer or CTO in-house (A, 162; B, 126; B, 239-240; G, 149-152). Participant C believes that the utilisation of No-Code solutions reduces the technical manpower needed in the founding team or core team (C, 244-245). D reveals the beginnings of the startup he is working for and mentions that the initial version of their website was created by a non-engineer (D, 422), meanwhile the company received a Series B funding. Technical abilities are not mandatory but increase efficiency when utilising No-Code solutions (A, 170-174; D, 385-387; G, 122-124). This statement is predominantly true for simple applications, such as standard e-commerce platforms (G, 149-152). In the case of a startup founder, it is possible to eliminate the need for a developer by utilising AI-driven No-Code solutions. It may take the founder some time and many attempts, till a desired result is obtained but it is possible even with lack on technical expertise (E 132-138). Technical skills are not essential in the beginning, but at a later stage, they are indispensable when complexity increases (E 208-210). Participant E argues that the technical component can be dispensed within the initial phase. However, there comes a time when someone with adequate technical skills must take over. Therefore, only a shift in the required competencies takes place. According to him, it is still necessary to have the same people as before (E, 198-202). A similar opinion is held by participant I saying

> *when people with no technical skills build tools on Bubble because of their poor architectural choice in the beginning, there are a lot of problems down the road, that may stop them from scaling. So ideally, at least in the beginning, you would have someone to, at least, do the basics […]* (I; 162-165).

However, not only poor decisions at the beginning can be obstacles. The entrepreneurial venture of D started facing challenges when processes became more complex and too many people without sufficient technical understanding made modifications, resulting in errors infiltrating the system. Therefore, the software developers assist in instructing on the use of the tools to new employees and employees with limited technical abilities. Their number of software engineers remains low, as attempts are made to empower employees (D, 385-390). Interviewee A believes that the greatest benefits can be achieved if No-Code users are able to code. With this team composition, the benefits of No-Code solutions can be leveraged and yet there is an awareness about limitations (A, 170-174).

Competencies that continue to be in demand

The majority of participants explicitly state that software development skills continue to be relevant (A, C, E, F, H, I, J, K, L). C believes that everyone utilizing No-Code platforms ends up using Low-Code platforms and therefore requires knowledge about coding (C, 65-66). It opens a wide range of possibilities, nevertheless, understanding the underlying logic is still needed (A, 147-148, 150-151). Even the interviewed COO of a No-Code platform states that

> *No-Code doesn't substitute us having a technological expertise or an expert, a specialist on board in the long run. But it helps us to get started, whether we are businesspeople without technological background or if we are technology founders without business background* (F, 221-224).

K implies that from Series A to the latest Series B, it is crucial to have a CTO in place (K 16-18). The venture capital partner investor J expresses his concern that:

> *If someone believes, that using No-Code tools means that there is a sort of shortcut, it's really dangerous, in my opinion. If someone is aware and knowledgeable about what he is doing, I would not be significantly worried* (J, 68-70).

He illustrates his quote with a metaphor that says someone should not assume to be an AI expert and not be able to solve the simplest addition or subtraction task without a calculator and solely relying on a machine (J, 68-72). However, the use of No-Code technologies offers a significant time-saving potential, which allows to focus on competencies gaining in relevance (G, 134-135), shown subsequently.

Competencies with increasing significance

It should be mentioned that the experts indicate competencies that are gaining in importance and some positions are expanding in scope. The skills explicitly mentioned are expertise in design, especially usability design and interface design as well as marketing and sales. Product manager scope of activity will increase and problem-solving and being self-driven is considered a core competence (B, 240-241; D, 351 & 356-360; E, 172-177). The role of product managers

undergoes a transformation according to E, as they are required to acquire expertise in the utilisation of No-Code solutions. This enablement empowers product managers to autonomously build prototypes to showcase their innovative ideas. In the initial phase of the creation of the prototypes, the involvement of software developers is not needed. Once the prototype is validated, the involvement of a developer becomes crucial. The prototype developed by a product manager does not have to be technically mature, since this is the task of the software developer. The involvement of the software developer is postponed until later, yet offers the benefit, that the developer can rely on a functional prototype that only needs to be refined and optimised. Faster development due to more comprehensive work instructions for the software developer are the outcome (E, 172-177).

5 Discussion

Within this section, the findings from chapter 4 will be related to the previously elaborated theory from chapter 2. Therefore, the derived categories and subcategories illustrated below serve as a structural guiding line to answer research question one and two in detail and thus, contribute to the current literature.

Category	Subcategory	Relation to Research Question (RQ)	
Factors influencing the acceptance	Drivers for No-Code utilisation	RQ 1	
	Inhibitors for No-Code utilisation	RQ 1	
State-of-the Art software development approach across the stages of startup development	No-Code as adequate approach	RQ 1	RQ 2
	Traditional coding as adequate approach	RQ 1	RQ 2
Future Development of No-Code technology	No-Code and AI	RQ 1	RQ 2
Impact of No-Code Movement on Workforce	Reduction of workforce		RQ 2
	Shift of essential competencies within a core team		RQ 2

Table 5: Overview of categories and relation to research questions
Source: Own illustration

5.1 Which factors influence the utilisation and acceptance of No-Code solutions? (RQ1)

There are several driving and inhibiting factors to be found within the current literature, as well as assessments regarding under which circumstances Low-Code/No-Code solutions are suitable and thus shall be applied. Subsequently, starting with drivers, categories and subcategories are separately discussed and displayed, whether and how the literature matches with the empirical findings.

Drivers		Inhibitors	
A	Speed of Development	M	Vendor lock-in
B	Rapid validation and testing of a potential business idea	N	Limited customization and flexibility
C	Easy to learn	O	Security concerns: lack of On-Premises support
D	Developer scarcity	P	Complex requirements towards solutions
E	Financial gain through usage of No-Code solutions	Q	Insufficient maturity level of No-Code technology
F	No-Code solutions get problems solved	R	Lack of understanding of benefits obtained by No-Code solutions
G	Merge of AI-Trend with No-Code-Trend leads to higher development efficiency	S	Lack of possibility for scaling
H	Integration in other tools	T	Lack of experience and trust
		U	Missing ownership of Intellectual Property and the source code

Table 6: Drivers and inhibitors identified in interviews
Source: Own illustration based on interviews and literature (Brühl et al., 2023; Sufi, 2023)

5.1.1 Drivers for No-Code utilisation

Throughout the whole research process, the following drivers were found to be important: A) Speed of development, B) Rapid validation and testing of a potential business idea, C) Easy to learn, D) Developer scarcity is addressed, E) Financial gain through lower development costs, F) No-Code solutions get problems solved, G) Merge of AI-Trend with No-Code-Trend leads to higher development efficiency and H) Integration in other tools. Subsequently, each driver will be discussed specifically.

A) Speed of Development:

Development speed is seen as an important driver for the usage of No-Code solutions by many interviewed experts and in the literature e.g., by Sufi (2023), Brühl et al. (2023) and Lu (2020). Through utilizing these solutions and thus obtaining a higher level of development speed, time-to-market can be reached faster (B, C, G, H). Time-to-market is especially critical in early stages, since competitors attempt to enter the market once the business idea is validated (A, 38). Therefore, development speed is an important driver. B, C, G and H mention, that using No-Code solutions leads to more rapid iterations and a higher level of flexibility, since one can accomplish more within less time. In their view, this results in competitive advantage and substantial cost savings. D states, that coding a specific task with a traditional coding approach normally required in between three and five days, whereas now, the same task, by using No-

Code solutions, can be conducted within half a day (D, 78-79). Accelerated software development through No-Code usage is therefore an implication delivered by D, being in line with Rymer (2018) and Trigo et al. (2022). The latter mention, that the speed level is three times higher, however, Rymer (2018) states that it is five to ten times faster than the traditional way of coding. Therefore, a waste of time is avoided by creating codes with already available "Lego bricks" (E, 81-83) codes and states, that missing bricks can be added by an individual established and incorporated code.

B) Rapid validation and testing of a potential business idea

The possibility to rapidly validate and test a potential business idea with No-Code solutions (K) and this without the need for high investments is an important driver, as stated by Sufi (2023). This is confirmed by H and I mentioning that there is less capital required initially for validating ideas, resulting into faster launches of businesses and the possibility to generated money directly.

C) Easy to learn

Furthermore, No-Code solutions can be easily learned (Sufi, 2023) and easy to be used (Brühl et al., 2023), which means, according to D, that anybody capable of dealing with No-Code solutions or junior developers can fulfil a task without the need for a startup of a costly senior developer. Thus, according to H and I, the need for an early investment is postponed and as a result, lower entry barriers for founders using No-Code solutions, are generated. These lower entry barriers for launching a business have been also stated by Kass et al. (2022) and were confirmed by experts.

D) Developer scarcity

By applying No-Code solutions, entrepreneurs are capable of tackling the dilemma of developer scarcity, which is existent on the labour market. This need for inhouse developers is reduced, according to G. According to the author, here can be drawn a connection to driver C), also knowing D's conclusion, that a core team, consisting of senior developers, is less important, than it used to be before having the possibility of using No-Code solutions (D).

E) Financial gain through usage of No-Code solutions

A driver, making the utilisation of this technology very attractive, is the result of lower development costs (Brühl et al., 2023), since high costs for senior developers do not need to be

spent as before, also confirmed by B, D, H, I. As a consequence, the author draws the connection towards driver B), confirming, that less capital is required to validate and test business ideas and driver C) with a postponed requirement for an investment. For small sized ventures, D recommends the usage of No-Code solutions in all stages for cost-performance reasons.

F) No-Code solutions get problems solved

No-Code solutions do, what needs to be done by an entrepreneur through application of fast, innovative solutions, which by using, shows an entrepreneurs' *"can-do-attitude"* (F) by solving problems hands-on. However, whether and when No-Code solutions are adequately suitable for a special case, will be discussed in a subsequent chapter. Here, the author draws a line with Sufi (2023), who mentions, that No-Code technology is highly adopted through its advantage of easy usage (Driver C) and rapid development (Driver A) and its simplified way of developing. In the end, more applications can be developed with No-Code solutions, than it would have been possible, when only applying the traditional approach (F). No wonder, that in some areas, like the development of websites, No-Code solutions are seen already as standard (K).

G) Merge of AI-Trend with No-Code-Trend leads to higher development efficiency

AI-tools, which enhance platforms based on No-Code technology lead to a higher development efficiency, which can be seen as an important driver in Sufi (2023). F sees a merger of the trend of Artificial Intelligence and the trend of No-Code happening, being supported by I and E. F thinks that within circa three to five years, there's no longer a distinction existent anymore in between the two. Being supported by generative AI to create No-Code apps, in his view, it is possible to create these apps within 18 to 60 seconds (F). Furthermore, he mentions, that generative AI will have an even more dominant role, which does not hinder him to believe in No-Code solutions.

H) Integration in other tools

As mentioned by Sufi (2023), No-Code run platforms are now integrated into other platforms and tools like a database or a cloud platform and thus, enable the set-up of applications, which possess a higher complexity level. Regarding the future development, interviewee I thinks, that vendors selling No-Code solutions will include assistants in order to speed up (driver A, comment by the author). Nevertheless, this degree of integration depends on the costs that come along with it. On one hand, the integration process is expensive, on the other hand, integration improves the developers productivity level by 50% and thus being an important driver (I).

As demonstrated above, there are many factors, that support an utilization of No-Code solutions. Nevertheless, for an adequate assessment of the technology, one must also consider existent negative influencing, inhibitor factors that come with this technology. These inhibitors out of literature and experts point of view are described in the subsequent chapter.

5.1.2 Inhibitors for No-Code utilisation:

M) Vendor lock-in

Once, a developer has decided for a specific No-Code platform, and initiates to build its solution on it, he/she is stuck to the vendor's platform; in other words, the developer becomes dependent on the platform and is subject to a vendor lock-in (Beranic et al., 2020; Brühl et al., 2023; Luo et al., 2021). This issue is also seen by expert B. In case of moving to another platform in later stages, the process regarding the development and design must be done once more for the whole solution, since an adequate integration in between the existent No-Code platforms on the market is not given yet, rather each platform provides an own ecosystem (Sufi, 2023). Furthermore, in the eyes of C, Investors are very hesitant to invest in solutions based on No-Code, since outsourcing of core technology and thus, reliance on external providers is still negatively perceived by them.

N) Limited customization and flexibility

According to Kass et al. (2023) and Brühl et al. (2023) No-Code technology provides entrepreneurs with a limited level of customization and flexibility. The latter is also mentioned by Sufi (2023). Brühl et al. (2023) even add limited functionality. The author of this thesis subsumes this under lack of technological flexibility and functionality. E confirms this lack of technological flexibility by stating that unique products shall not be created with No-Code solutions, since in almost any case, they provide a previously built standard-kit (E). This thesis' author concludes that standard-kits and a requirement for unique solutions are mutually exclusive. As a finding, flexibility in the context of No-Code solutions has to be seen two sided. First, it can be positive when it is time-related, meaning when it comes to providing speed and thus, allowing to do more in less time as shown in driver A. Second, when it comes to the technical perspective in developing, a limited flexibility can be assessed, when utilizing No-Codes solutions.

O) Security concerns: lack of On-Premises Support

Almost all on the market available No-Code platforms apply a cloud-based approach (Sufi, 2023). The No-Code created algorithms so are always online. However, in some cases, an offline approach is absolutely required. For instance, intelligence or military solutions, as well as secret health data from patients or government data must be offline and thus exclude No-Code as a choice (Sufi, 2023), which is confirmed by F. C, I and E mention regulatory requirements for e.g., hosting data, as a reason. F even goes further, saying, that their usage must be seen critically in industries with a high level of regulation, where dealing with sensible data is part of daily life. He summarizes, that the utilisation should be cautiously assessed when one is dealing with identity, data security and data protection. Nevertheless, he says, using No-Code solutions also in these industries is possible but often does not suit the requirements best.

P) Complex requirements towards solutions

It also can be stated that No-Code solutions are not the perfect fit, when highly complex solutions are demanded, or differently said by A, *"[a]s complexity increases, suitability for No-Code solutions decreases*" (A, 60-64). And when E adds, that No-Code solutions offer a previously built kit in most cases, the thesis' author concludes, that this standard-kit does not stand the requirements towards highly complex solutions. This conclusion is also agreed by E's point of view, expressing that No-Code technology shall not be utilised when there is a need for unique solutions. The decision for or against a startup is made in the persuasion stage. Complexity as a decisive factor in this stage plays an important role in Rogers (1983; 2003) Innovation-Decision Process Modell, shown in the characteristics of innovation in chapter 2.2.1. Thus, highly complex requirements can be seen as an inhibiting factor for No-Code solutions, resulting in a potential decision against a startup relying on this technology within the persuasion stage.

Q) Insufficient maturity level of No-Code technology

The current maturity level of No-Code is perceived to be not as advanced enough, as it should in order to be called sufficient, according to E, J and D. That's the reason, why K see's it critical, when a startup relies on external providers and thus source important core technology out to them. According to him, outsourcing can be done partially once the technology can be called sufficient. However, B does not perceive the actual maturity level to be insufficient and therefore see's it not as an inhibitor for investing in or utilising a startup application based on

No-Code technology. F supports B by mentioning that vendors of No-Code solutions have done an eminent advancement in order to be perceived as reliant actors.

R) Lack of understanding of benefits obtained by No-Code solutions

Another inhibitor, as assumed by B, is that some founders do not possess the understanding regarding the benefits obtained through the usage of No-Code solutions, namely: low costs for development and speed. The author combines that when the drivers A) and E) are not adequately understood, this missing knowledge about advantages is an inhibitor for the usage of No-Code solutions.

S) Lack of possibility for scaling

Absence of scalability is an inhibitor mentioned by Kass et al. (2022), that leads to a less successful adoption of No-Code technology. So, if one's target is to set up a new company like Facebook, scaling is necessary therefore. To be able to scale, a transition to coding has obligatory to be done, according to I and K. In their eyes, essentially important components cannot be realistically outsourced, having the target of scaling in front of you.

Furthermore, some **inhibitors** shall be mentioned, targeting especially **the investors point of view**. Therefore, inhibitors T) and U) are shown subsequently.

T) Lack of experience and trust

For some investors No-Code is actually perceived rather a liability than an asset for investors, when following A's argument. This is, in his eyes, mainly the result of lacking experience. Also professionals with a background in computer sciences think critically about No-Code solutions (D).

U) Missing ownership of Intellectual Property of the source code

Brühl et al. (2023) perceive it as negative influencing factor, that an entrepreneur who builds an application with No-Code solutions does not possess the Intellectual Property (IP) ownership of the underlying source code. B and J also see this concern, underlining, that from an investors perspective, the red flags, resulting from the lack of possessing the IP-ownership, lie in the areas of legal and business risks. That is, where a harmful influence can come. Further B provides two scenarios. First, in case the No-Code vendor is not successful anymore at any time in the future, the fundamental basis is lost and thus, a high dependency on the vendors success

is given. Second, he states that, entrepreneurs using a vendors' No-Code solution cannot adequately calculate whether there will be a change in the vendors licence – pricing, which leaves the uncertainty, whether an entrepreneur with these risen prices is able to generate profit and is thus furthermore a sustainable working startup.

Having heard all the driving and inhibiting factors, in the eyes of the author, the decision for or against a startup using No-Code solutions strongly depends on an entrepreneurial founder's goal with its startup and whether a special No-Code solution fits this goal and therefore suits adequately. Subsequently, it will be discussed, under which circumstances, No-Code solutions are the adequate option, and under which condition traditional coding is more suitable.

Adequate coding approach: Traditional vs. No-Code

No Code as suitable approach

Stage-related: The experts A, B, C, D, F, G, H, I, J, and K possess the opinion that No-Code solutions are relevant within all startup stages (shown in chapter 2.1.2). With the Munich based startup FINN, A provides an example, where this is applied in practice. In contrast, E underlines, that No-Code solutions meet the requirements just in the initial phase adequately, when developing the product and recommends later, once customers show willingness to pay for the product, a transformation to first, Low-Code and then to custom code. However, the interviewees declare, that No-Code solutions have the highest level of relevance in the early stages. This is when founders are looking whether a market is given for a solution, i.e., product market fit is being considered, and whether a solution is able to solve the customers problem. Testing this, driver B) is identified by the author as being affective in this phase. For D, acting fast in this phase is highly important and No-Code makes that possible. Also, L believes, that *"[d]uring pre-acceleration idea stage, No-Code is a must" (*L, 83).

Complexity-related: No-Code solutions also perfectly fit the demands, when simple rules, such as if-then or notifications or alerts are required, according to D. H sees websites or software application as areas, where No-Code solutions can be the adequate choice. E thinks, that No-Code is especially a good fit for small ventures with the goal of automation or also generally everywhere where automation needs to be done.

Cost-related: For cost-based reasons, D recommends small businesses to further continue with No-Code solutions due to the cost-performance ratio. He further mentions that the size of a startup is highly crucial.

Furthermore, it will be referred to the drivers above and an entrepreneurs' choice.

No Code as unsuitable approach

The suitability for No-Code solutions decreases, when highly complex solutions are required (A). Furthermore, G states that No-Code solutions are not particularly suitable for highly innovative applications and neither, as E says, when an entrepreneur has the goal of creating a unique solution due to a No-Code standard-kit. C, E, I and F also state that regulated industries are not an area, where No-Code solutions perfectly can be used, as demonstrated in inhibitor O) above. Especially when looking at critical processes such as the money-making one, D recommends custom coding, in order to ensure the traceability and in order to find errors fast.

Change of approach – Transformation to traditional coding

It shall be mentioned that in later stages, entrepreneurs' priorities shift (F, H). In their eyes, in contrast to the initial phase, where No-Code was the core element for creating value, in later stages, this technology becomes more a supportive character. H identifies that, at the latest in the maturity stage, the earlier focus on speed is not the main focus anymore. In the maturity stage, the new focus lies rather on optimising the validated product in order to obtain the best version of the product. Also, he recommends switching to coding, once system-based limitations appear and D suggests the change, once product market fit is reached. According to him, at the latest, a decrease of usage of No-Code solutions happen in Series A, or latest in Series B, when system-based limitations appear (D).

E states, that No-Code is just suitable for the initial phase and must be obligatory changed, recommending first, to move from No-Code to Low-Code and then to code. The time for this change shall be reached, once customers are willing to pay for a solution, according to him. Participants F and G also see that the usage of No Code-based solutions in later stages is getting more specialised and thus decreases a little. Further, I and K declare a transition to code inevitable in case an entrepreneur's goal is the scalability of its application.

All in all, No Code solutions offer *"logic as a service" (*E, 94-96) and represent a basis for developing of knowledge in this field, regarding individuals with or without technical background (F). In the end, depending on the individual goal towards an to be developed application, each entrepreneur needs to assess the drivers and inhibitors adequately and based on this, take an informed decision.

5.2 How does the existence of No-Code solutions change the team composition of startups in different stages? (RQ2)

The in-depth investigation on how a team composition changes due to the influence of No-Code solutions resulted in the subsequent displayed categories which are based on the four workforce-roles, every startup needs. The roles are explained in chapter 2.1.3.

Changing demand for the workforce group "hackers"

The previously dominant determent technical ability, utilised when assessing a teams' quality, has lost some importance, since a wide range of tasks can be also conducted by using No-Code solutions and thus, the ability to code/programme is not anymore considered to be a critical element, according to A, B and G. Nevertheless, technical abilities are not essentially obligatory when working with No-Code solutions, but enhance the teams efficiency (A, D, G). The truth of this statement is confirmed by G pointing out that it applies for simple applications. Lu (2020) observes, that nowadays, more and more non-technical people are founding their company and instead relying on developers they rely on No-Code solutions. But as E states, initially, technical skills are not a must-have, but they become one in later stages because of the then appearing complexity. Therefore, only a shift in the required competencies takes place.

Following B, the development costs in the future will decrease immensely. In his eyes, the amount of demanded developers then can be estimated with ten percent of the current amount, since one can fulfil the task, which actually requires a team of ten. According to C, the demand for mid-level developers in future will sink and explains this phenomenon with new, available technology on the market, such as ChatGPT or LLM, which perform even better than humans. AI-generated No-Code solutions provide the opportunity for startups to replace developers and spare them (E). However, E opposes C's opinion by stating that mid-level developers rather will be further demanded, since they cost less than senior developers, but offer adequate experience.

Rising demand for the workforce-group "design skills"

Design skills are increasingly demanded, particularly in expertise about interface- and usability-design. Further, skills in the area of sales and marketing are in rising demand (B, D, E).

Workforce-group "visionary"

A change due to the influence of No-Code solutions has not been mentioned by experts. For the authors, this makes sense, because the group of visionaries by definition occupy themselves with a ventures' strategy and its long-term goal. No-Code solutions therefore do not alter or modify the conditions of this group as strong as the others within a startup.

Workforce-group "hustlers"

Product managers possess the characteristically attributes of Hustlers as seen in chapter 2.1.3. Their importance and range of activity is going to expand, according to B, D and E. They also see core competencies in someone, that is self-driven and capable to solve problems. Nevertheless, according to E, the product managers role will be modified, since they need to gain knowledge in how to use No-Code solutions and thus are also capable of building a prototype on their own and so, showing their innovative way.

6 Conclusion

Within this thesis, important factors driving or inhibiting the usage and acceptance of No-Code solutions and the technology's influence on a founder's team composition have been investigated. Therefore, empirical information from expert interviews were collected and an in-depth analysis conducted, evaluating transcripts with Maryings' qualitative content analysis. The author provides a detailed list of erased positive and negative factors and their impact on an entrepreneur, when using No-Code solutions. Even when adequate areas of application are recommended, each entrepreneur has to take its own decision for or against the usage of No-Code solutions, since its adequacy is dependent on the entrepreneur's individual goal(s). The results provided, shall serve as a detailed information base to take an adequate decision.

For the author, a central finding is to understand, that an entrepreneurs' priorities strongly shift over time, depending on its startup-stage and that this focus has an immense influence on whether to use No-Code solutions or not at a time (F, H). Therefore, one must know, that a fast development is highly critical to success in early stages, since competitors can enter the market once a business idea is validated (A). Thus, the derived driver A) Speed of Development, is eminently important in early stages and loses later on some importance e.g., in the maturity stage, when optimisation of the validated product, in order to have the best version of it, is the new priority. Nevertheless, at the beginning, speed of development is critical and No-Code solutions provide it. By using No-Code solutions, in between three to five days, compared to traditional coding of the very same task can be saved, according to expert D. This is in line with the literature, having a range of three to ten days. Further, business ideas can be rapidly tested and validated without high investments, resulting in rapid business launches, the possibility to earn money directly, conducting fast iterations and generally, provide time-flexibility, since more can be accomplished in less time (D). Additionally, driver C) demonstrates, that because of its easy way to learn and usage of No-Code solutions, anybody capable of working with it or junior developers can fulfil tasks. As a result, costly senior developers can be spared and thus, high investments postponed (D; Brühl et al., 2023; Sufi, 2023). The financial gain due to the fact, that senior developers do not need to be engaged as before, replacing some through usage of No-Code solutions is another driver (D, H, I, B) This also solves the dilemma of developer scarcity, driver D). Also the fact, that No-Code solutions get problems solved, in other words, it does what needs to be done in an innovative way, is driving the usage (F). Finally, more applications can be developed by using this simplified way of coding than it would have been possible the traditional way. So it's not surprising that in some areas like website development,

No-Code is already standard (K). Even when integration processes are costly, it rises a developers level of productivity by 50%, as said by I. This integration of No-Code platforms in other platforms or other tools such as databases are momentarily taking place. Applications, then built with it, possess a higher level of complexity, according to Sufi (2023).

The impact of the present merge of the trends Artificial Intelligence (AI) and No-Code is another finding. AI-tools can improve platforms based on No-Code technology, which leads to a higher development efficiency (Sufi, 2023). For instance, supported by generative AI, a No-Code app can be built within 18 to 60 seconds (F).

Nevertheless, there are also reasons, why entrepreneurs avoid using No-Code solutions. One important reason is that an entrepreneur gets dependent on a vendors' No-Code platform once decided to work with it, which is called vendor lock-in (B; Beranic et al., 2020; Brühl et al., 2023). This is dangerous, since in the case of switching to another platform in a later stage, the process of designing and development must be done again, since each platform has its own ecosystem and an adequate integration in a new platform is not given yet (Sufi, 2023). No-Code solution furthermore just provide a limited level of customization, technical flexibility (Brühl et al., 2023; Kass et al., 2022) and functionality (Brühl et al., 2023), since in almost all cases, they just offer a standard-kit, which does not allow to create unique applications (E). As conclusion a standard-kit does not allow to build highly complex solutions. Or as A describes inhibitor P): *"[a]s complexity increases, suitability for No-Code solutions decreases"* (A 60-64). Their usage must be also critically seen in highly regulated industries, when one must fulfil strict regulatory requirements, like the ones for data hosting. Since almost all available No-Code platforms offer a cloud-based approach and these industries operate with sensible data, it's an exclusion criterion for intelligence, military, or health data applications, where data must be offline hosted. Another finding is, that some experts (E, D, J) consider the maturity level of No-Code solutions insufficient and that in this stage, entrepreneurs should not do a technology outsourcing (K), while others perceive an important technology advancement (B) which can be relied upon (F). Furthermore, the absence of scaling possibility is inhibitor (Kass et al., 2022). If one wants to follow this goal, an obligatory change to coding must be done (I, K). From an investors' perspective, the missing IP-ownership of the source codes must be seen as legal and business risk (B, J), since an entrepreneur is dependent on a No-Code vendors success and is subject to uncalculatable changes in license-pricing. To conclude, No-Code solutions are relevant in all startup stages, as said by almost all experts. Nevertheless, the highest level of relevance is given in the early stages and especially suitable for not too complex applications.

But how do No-Code solutions affect the composition of a startup's team?

As seen in chapter 2.1.3, a startup needs four different character groups in the team: "hackers", "designers", "visionaries" and "hustlers". These groups underlie a different impact by No-Code solutions. While an effect on the visionary group has not been mentioned by any expert, the other groups are subject to changes. The concluding finding is, that visionary people by character definition are occupied with business strategy, which is not targeted by No-Code solutions. Therefore, this group is not subject to changes. However, the determining factor technical ability, the ability to code has lost some importance (A, B, G) since No-Code solutions can also provide it. This affects "hackers" the most. While one experts' side (B, C) thinks, that future demand for developers will sink; e.g., in B's eyes there will be just a demand for ten percent of the current amount, E's view is, that developers are further demanded and that there will be just a shift in required competences. He explains that technical skills are not a must-have initially, but once it becomes more complex in later stages, it becomes one. The demand for both "design skills" and "hustlers" will rise as a result of No-Code technology, since expertise, particularly in interface- and usability-design, marketing and sales for the first group is desired (B, D, E). The demand for the latter group will increase, since problem solvers, self-driven people are needed. Also the groups field of acting will alter. Product managers will gain knowledge in No-Code technology, apply it and thus contribute in an innovative way e.g., by building a prototype on their own.

Finally, depending on one's individual goal, an entrepreneur must decide, whether No-Code solutions as *"logic as a service"* (E, 94-96) meet that goal. To take an informed decision, the author provides in-depth information about drivers, inhibitors, suitable areas of application and the change of a startups team composition.

7 Limitations & Future Research

This thesis contains data gained through expert interviews and implemented the qualitative content analysis method, which perfectly suits the target of investigating research question one and two.

Nevertheless, this study is also subject to limitations. First, the number of experts participating in the interviews is limited to 12 participants. Potentially, the emerging results and thus, the argumentation could have been enhanced through a higher number of experts or further target groups. Another limitation is the authors' primary focus on the early stage, while secondary looking at later stages.

Future research is needed in order to obtain a higher level of knowledge regarding the two research questions. Therefore, the author suggests that further research should mainly focus on other startup stages or modify the composition of the expert group.

Bibliography

Adkin, D. (2020). *The future is no-code.* Adalo. Retrieved Jul 21, 2023, from https://www.adalo.com/the-future-is-no-code/conclusions

al Alamin, M. A., Malakar, S., Uddin, G., Afroz, S., Haider, T. B., & Iqbal, A. (2021). An empirical study of developer discussions on low-code software development challenges. *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*. https://doi.org 10.1109/MSR52588.2021.00018

Alemany, L., & Andreoli, J. (2018). *Entrepreneurial finance: The art and Science of Growing Ventures*. Cambridge University Press.

Beranic, T., Rek, P., & Heričko, M. (2020). Adoption and usability of Low-Code/No-Code Development Tools. https://search.proquest.com/docview/2531366275

Blank, S. (2011, Dec 13,). *The Startup Team.* Retrieved Oct 19, 2023, from https://steveblank.com/2011/12/13/the-startup-team

Blank, T. H., & Carmeli, A. (2020). Does founding team composition influence external investment? the role of founding team prior experience and founder CEO. *The Journal of Technology Transfer, 46*(6), 1869-1888. https://doi.org/10.1007/s10961-020-09832-3

Brocoders. (2021, March 6). *The pros and cons of low-code development.* Retrieved Aug 17, 2023, from https://hackernoon.com/the-pros-and-cons-of-low-code-development-4y2p33g9

Brühl, S., Bernsteiner, R., Ploder, C., Dilger, T., & Spiess, T. (2023, May). The Use of No-Code Platforms in Startups. In *International Conference on Knowledge Management in Organizations* (pp. 289-301). Cham: Springer Nature Switzerland.

Bubble. (2023, April 6,). *Upcoming changes to bubble's pricing plans in 2023.* Bubble. Retrieved Aug 10, 2023, from https://bubble.io/blog/2023-pricing-updates

Carpentier, C., & Suret, J. (2015). Angel group members' decision process and rejection criteria: A longitudinal analysis. *Journal of Business Venturing, 30*(6), 808-821. https://doi.org/10.1016/j.jbusvent.2015.04.002

CB Insights. (2021, Aug 3,). *The top 12 reasons startups fail.* CB Insights. Retrieved August 14, 2023, from https://www.cbinsights.com/research/report/startup-failure-reasons-top

Computer History Museum. (2023). *Birth of the computer.* Retrieved Oct 16, 2023, from https://www.computerhistory.org/revolution/birth-of-the-computer/4/78

Computer Science Degree Hub. (2022, December 2). *What is a generation computer language and how is it used?* Computer Science Degree Hub. Retrieved Oct 29, 2023, from https://www.computersciencedegreehub.com/faq/what-is-a-second-generation-programming-language

Crowne, M. (2002). Why software product startups fail and what to do about it. evolution of software product development in startup companies. IEEE International Engineering Management Conference, 338-343. https://doi.org/10.1109/IEMC.2002.1038454

Dissanayake, C. A., Jayathilake, W., Wickramasuriya, H. V., Dissanayake, U., Kopiyawattage, K. P., & Wasala, W. M. (2022). Theories and models of technology adoption in agricultural sector. *Human Behavior and Emerging Technologies, 2022*, 1-15. https://doi.org/10.1155/2022/9258317

Döring, N., & Bortz, J. (2016). *Forschungsmethoden und Evaluation in den Sozial- und Humanwissenschaften* (5th ed.). Springer.

Dushnitsky, G., & Stroube, B. K. (2021). Low-code entrepreneurship: Shopify and the alternative path to growth. *Journal of Business Venturing Insights, 16*. https://doi.org/10.1016/j.jbvi.2021.e00251

Evans Data. (2022, Dec 13,). Number of software developers worldwide in 2018 to 2024. Statista. Retrieved Jul 31, 2023, from https://www-statista-com.eu1.proxy.openathens.net/statistics/627312/worldwide-developer-population

Feder, G., Just, R. E., & Zilberman, D. (1985). Adoption of agricultural innovations in developing countries: A survey. *Economic Development and Cultural Change, 33*(2), 255-298. https://doi.org/10.1086/451461

Formstack. (2021, April 18,). *What problems were you trying to solve with no-code tools?* The Rise of the No-Code Economy. Retrieved Jul 20, 2023, from https://resources.formstack.com/reports/rise-of-the-no-code-economy/tech-revolution

Frampton, D., Blackburn, S. M., Cheng, P., Garner, R. J., Grove, D., Moss, J. E., & Salishev, S. I. (2009). Demystifying magic: High-level low-level programming. In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, 81–90. https://doi.org/10.1145/1508293.1508305

Ganor, Y. (2022, January 14). Ask an expert: What skills do I need to run a startup? Harvard Business Review. Retrieved October 19, 2023, from https://hbr.org/2022/01/ask-an-expert-what-skills-do-i-need-to-run-a-startup

Gartner. (2022, Dec 13). *Gartner forecasts worldwide low-code development technologies market to grow 20% in 2023.* Gartner. Retrieved Jul 20, 2023, from https://www.gartner.com/en/newsroom/press-releases/2022-12-13-gartner-forecasts-worldwide-low-code-development-technologies-market-to-grow-20-percent-in-2023

Giardino, C., Bajwa, S. S., Wang, X., & Abrahamsson, P. (2015). Key challenges in early-stage software startups. In *Agile Processes in Software Engineering and Extreme Programming: 16th International Conference, XP 2015, Helsinki, Finland, May 52-63, Proceedings 16* (pp.52-63). Springer International Publishing. https://doi.org/10.1007/978-3-319-18612-2_5

Giardino, C., Unterkalmsteiner, M., Paternoster, N., Gorschek, T., & Abrahamsson, P. (2014). What do we know about software development in startups? *IEEE Software, 31*(5), 28-32. https://doi.org10.1109/MS.2014.129

Gläser, J., & Laudel, G. (2009). *Experteninterviews und qualitative Inhaltsanalyse* (3rd ed.). VS Verlag.

HP. (2018, Oct 15,). *Computer history: A timeline of computer programming languages.* Retrieved Oct 17, 2023, from https://www.hp.com/us-en/shop/tech-takes/computer-history-programming-languages

Hughes, A. (2023, Mar 29,). *The software developer is dead: Long live the software developer.* Forbes. Retrieved Aug 3, 2023, from https://www.forbes.com/sites/forbestechcouncil/2023/03/29/the-software-developer-is-dead-long-live-the-software-developer

Hurlburt, G. (2021). Low-code, no-code, what's under the Hood? *IT Professional, 23*(6), 4-7. https:doi.org/10.1109/MITP.2021.3123415

IBM Cloud Education. (2022, May 23,). *Low-code vs. no-code: What's the difference?* Retrieved July 31, 2023, from https://www.ibm.com/blog/low-code-vs-no-code

iCIMS. (2019, Oct 8). *Employers hired only 60% of their required tech talent in the U.S. over the last three years, based on iCIMS' analysis of 25 million applicants*. Retrieved Aug 8, 2023, from https://www.icims.com/company/newsroom/employers-hired-only-60-of-their-required-tech-talent-in-the-u-s-over-the-last-three-years-based-on-icims-analysis-of-25-million-applicants/

Jurkenas, R. (2023, Jan 17,). *Low code history*. Retrieved October 2, 2023, from https://codeornocode.com/no-code/low-code-history

Kaiser, R. (2021). *Qualitative Experteninterviews: Konzeptionelle Grundlagen und praktische Durchführung* (2nd ed.) Springer VS. https://doi.org/10.1007/978-3-658-30255-9

Kass, S., Strahringer, S., & Westner, M. (2022). Drivers and inhibitors of low code development platform adoption. *2022 IEEE 24th Conference on Business Informatics (CBI), 1.* https://doi.org/10.1109/CBI54897.2022.00028

Koster, A. (2019, Nov 8,). *How to prepare for tech hiring in 2020.* iCIMS. Retrieved August 8, 2023, from https://www.icims.com/blog/how-to-prepare-for-tech-hiring-in-2020

Krajewski, R. (2021, Oct 14,). *The rise of no-code and low-code solutions: Will your CTO become obsolete?* Forbes. Retrieved July 21, 2023, from

https://www.forbes.com/sites/forbestechcouncil/2021/10/14/the-rise-of-no-code-and-low-code-solutions-will-your-cto-become-obsolete

Kuckartz, U. (2018). *Qualitative Inhaltsanalyse. Methoden, Praxis, Computerunterstützung* (4th Edition ed.). Beltz Juventa.

Lu, E. (2020, April 1,). *The no-code movement: From technopreneur to entrepreneur.* Forbes. Retrieved August 3, 2023, from https://www.forbes.com/sites/forbesbusinesscouncil/2020/04/01/the-no-code-movement-from-technopreneur-to-entrepreneur

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). https://doi.org/10.1145/3475716.3475782

Mayring, P. (2015). *Qualitative Inhaltsanalyse: Grundlagen und Techniken* (12th ed.). Beltz.

Mayring, P., & Fenzl, T. (2014). Qualitative Inhaltsanalyse. In N. Baur, & J. Blasius (Eds.), *Handbuch Methoden der empirischen Sozialforschung*. Springer VS. https://doi.org/10.1007/978-3-531-18939-0

O'Regan, G. (2021). History of programming languages. *A brief history of computing*, 177-200. Springer.

Peterson, B. (2017, Oct 11,). *The CEO of GitHub, which caters to coders, thinks automation will bring an end to traditional software programming.* Business Insider Nederland. Retrieved Aug 13, 2023, from https://www.businessinsider.nl/github-ceo-wanstrath-says-automation-will-replace-software-coding-2017-10

Petty, J. S., & Gruber, M. (2011). "In pursuit of the real deal": A longitudinal study of VC decision making. *Journal of Business Venturing, 26*(2), 172-188. https://doi.org/10.1016/j.jbusvent.2009.07.002

Pham, T. (2021, Apr 13,). *Analyzing the software engineer shortage.* Forbes. Retrieved Aug 3, 2023, from https://www.forbes.com/sites/forbestechcouncil/2021/04/13/analyzing-the-software-engineer-shortage

Rafiq, U., Filippo, C., & Wang, X. (2022). Understanding low-code or No-code adoption in Software startups: Preliminary results from a Comparative case study. *Product-Focused Software Process Improvement*, 390–398. https://doi.org/10.1007/978-3-031-21388-5_27

Richardson, C., Rymer, J., Mines, C., Cullen, A., & Whittaker, D. (2014). New Development Platforms Emerge for Customer-Facing Applications. Forrester.

Rizwan, O. (2018, April). *A snapshot of programming language history.* Increment. Retrieved Oct 16, 2023, from https://increment.com/programming-languages/language-history

Rogers, E. M. (1983). *Diffusion of innovations* (3rd ed.). Free Press.

Rogers, E. M. (2003). *Diffusion of innovations* (5th ed.). Free Press.

Rogers, E., Singhal, A., & Quinlan, M. (2019). Diffusion of innovations. In *An integrated approach to communication theory and research* (3rd ed., pp. 415–434). Routledge. https://doi.org/10.4324/9780203710753-35

Ryan, F., Coughlan, M., & Cronin, P. (2007). Step-by-step guide to critiquing research. part 2: Qualitative research. *British Journal of Nursing, 16*(12), 738-744. https://doi.org/10.12968/bjon.2007.16.12.23726

Rymer, J. (2018, Aug 8,). Why you need to know about low-code, even if you're not responsible for software delivery. Forrester. Retrieved August 9, 2023, from https://www.forrester.com/blogs/why-you-need-to-know-about-low-code-even-if-youre-not-responsible-for-software-delivery

Sahay, A., Indamutsa, A., Di Ruscio, D., & Pierantonio, A. (2020). Supporting the understanding and comparison of low-code development platforms. In *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 171–178. https://doi.org/10.1109/seaa51224.2020.00036

Santalo, A. (2021, Nov 11,). *Why low-code platforms are the developer shortage solution people aren't talking about.* Entrepreneur. Retrieved Aug 17, 2023, from https://www.entrepreneur.com/science-technology/why-low-code-platforms-are-the-developer-shortage-solution/390569

Schwarzer, G. (2001). Forschungsanträge verfassen. Ein praktischer Ratgeber für Sozialwissenschaftler/-innen. *Zeitschrift für internationale Beziehungen, 8*(1), 141-156. https://doi.org/10.5771/0946-7165-2001-1-141

Seppänen, P., Oivo, M., & Liukkunen, K. (2016). The initial team of a software startup: Narrow-shouldered innovation and broad-shouldered implementation. In *2016 International Conference on Engineering, Technology and Innovation/IEEE Lnternational Technology Management Conference (ICE/ITMC)*. https://doi.org/10.1109/ice/itmc39735.2016.9026055

Smith, L. (2019, May 8,). *How the no-code trend will affect early-stage entrepreneurs.* Forward Partners. Retrieved Aug 3, 2023, from https://www.forwardpartners.com/latest/how-no-code-movement-going-impact-entrepreneurship

Sufi, F. (2023). Algorithms in low-code-no-code for research applications: A practical review. *Algorithms, 16*(2), 1-23. https://doi.org/10.3390/a16020108

Techbaz. (2023). *Programming language generations: 1GL, 2GL, 3GL, 4GL, 5GL, 6GL.* TechBaz. Retrieved Oct 29, 2023, from https://www.techbaz.org/Blog/Generation-of-programming-languages.php

Trigo, A., Varajao, J., & Almeida, M. (2022). Low-code versus code-based software development: Which wins the productivity game? *IT Professional, 24*(5), 61-68. https://doi.org/10.1109/MITP.2022.3189880

Tyebjee, T. T., & Bruno, A. V. (1984). A model of venture capitalist investment activity. *Management Science, 30*(9), 1051-1066. https://doi.org/10.1287/mnsc.30.9.1051

Unigram Labs. (2022). *The Low-Code / No-Code Ecosystem.* Medium. Retrieved August 16, 2023, from https://medium.com/@unigram_labs/the-low-code-no-code-ecosystem-8a0e0ba757c1

Vogel, R., Puhan, T. X., Shehu, E., Kliger, D., & Beese, H. (2014). Funding decisions and entrepreneurial team diversity: A field study. *Journal of Economic Behavior & Organization, 107*, 595-613. https://doi.org/10.1016/j.jebo.2014.02.021

Weichbold, M. (2014). Pretest. In N. Baur, & J. Blasius (Eds.), *Handbuch Methoden der Empirischen Sozialforschung* (pp. 299-304). Springer VS. https://doi.org10.1007/978-3-531-18939-0

Yarchevsky, M. (2021, August 16,). *Is a no-code platform right for your website or app project?* Forbes. Retrieved Aug 10, 2023, from https://www.forbes.com/sites/forbestechcouncil/2021/08/16/is-a-no-code-platform-right-for-your-website-or-app-project

Yin, R. K. (2018). *Case study research and applications: Design and methods* (6th ed.). SAGE.

Appendices

Drivers and inhibitors of Low-Code/No-Code adoption

	Driver		Inhibitor	
Relative Advantage	Improved performance metrics of the software development process	50		
	Improved working mode of the software development process	24		
	Improved output of the software development process	20		
	Reduced dependency on IT developer	19		
	Quicker reaction to market demand	7		
Compatibility			Difficulties to ensure interoperability with other LCDPs	14
			Difficult to integrate to other systems	13
			Limited of functionality of LCDPs	13
			Increased security, compliance, and privacy risk	11
			Reluctance to change behaviors	4
			Lack of governance structure	3
Complexity	Reduced entry barriers for application development	48	Lack of scalability	15
			Lack of flexibility and customization	15
	Ease to develop applications	35	Limited usability of LCDPs	8
			Lack of documentation	2
Trialability			Lock-in to a LCDP vendor	13
Observability			Difficulties in estimating total cost	9
			Fear of Shadow IT	2

Source: Adapted from Kass et al. (2022, p. 199)

Correspondence with Interviewees

Dear “Participant”,

Thank you for your quick reply on LinkedIn.

Enclosed is a brief overview of the interview, so you know approximately what to expect:

1. Introduction: Brief explanation in which field you work. (1-3min)
2. Familiarity: Your experience with No-Code solutions so far. (5-10min)
3. Usage: The usage of No-Code tools along the development stages (seed, startup stage, growth stage, maturity stage) of a startup. (5min)
4. Skillset: The impact of using No-Code tools on the composition of the founding team, especially on the skills required. (5min)
5. Investment Criteria: Determining factors for an investment in a young tech company using No-Code tools for their main product. (5-10min)

I would like to record the interview to transcribe it later. The recording will be deleted afterwards.
I hope you are okay with that?

Please feel free to directly book an appointment in my calendar that suits you:
https://calendly.com/florian-metzger/60min

If you have any further questions, I am at your disposal.

Best regards
Florian

Interview Guideline

Founders, No-Code vendors and investors receive the similar questions, but with tailored phrasing adapted according to the target audience.

Section: Introduction (3min)

1. Please can you briefly explain in which field you work & your experience with No-Code Tools?

Section: Experience with No Code (5-10min)

2. Can you please elaborate what the term No-Code means for you?
3. What are your reasons for using No-Code tools?
4. What factors do you use to assess whether No-Code is suitable for a project?
5. Under which circumstances would you advise against using No-Code tools?
6. Are there certain industries which are particularly suitable for the usage of No-Code tools?

Section: Usage according to Startup Stages (5min)

7. A startup goes through the phases seed stage, startup stage, growth stage & maturity stage. At which stages are No-Code tools relevant?
8. How does the use of No-Code tools change along the development stages (seed-, startup-, growth-, maturity stage) of a startup?

Section: Software Development Skills (5min)

9. What impact does the use of No-Code tools have on the software development approach in startup companies?
10. What impact has No-Code on the skillset needed in small teams of early-stage ventures? (If participant needs an explanation - e.g., management skills, design skills, technical skills – otherwise not mentioned)

Section: Investment Decision (5-10min)

11. Did you ever receive funding for you company when using No-Code tools to code the main product or solution?
12. What are the determining factors for an investment in a young tech company?
13. Are there factors that prevent investors from investing in a tech startup programmed exclusively based on No-Code solutions?
14. Do you see the use of No-Code as a driver or inhibitor when it comes to investment?