KU LEUVEN

FACULTY OF ECONOMICS AND BUSINESS

The Organizational impact of Low-code/No-Code on Agile Development Teams

Haikal Fikri

R0866511

Thesis submitted to obtain the degree of

MASTER OF BUSINESS ADMINISTRATION

Business Information Management

Promoter: Prof. Dr. Yves Wautelet

Academic year 2024-2025

KU LEUVEN

FACULTY OF ECONOMICS AND BUSINESS

The Organisational Impact of Low-Code/No-Code on Agile Development Teams

How does the use of Low-code/No-code platforms impact Agile development teams?

This research investigates the impact of Low-Code/No-Code (LCNC) platforms on Agile development teams, addressing a gap in existing literature that has primarily examined these technologies in isolation. Through semi-structured interviews with eight industry practitioners experienced in both LCNC platforms and Agile methodologies, this qualitative study explores how LCNC adoption affects team performance and dynamics within Scrum frameworks.

The study demonstrates that while LCNC platforms align well with Agile principles of rapid iteration and customer collaboration, their impact is highly context-dependent, varying with project complexity, team maturity, and management processes. These findings have practical implications for organizations considering LCNC adoption within their Agile frameworks, suggesting the need for careful consideration of project suitability and project management processes to maximize benefits.

Acknowledgements

I would like to thank Prof. Dr. Yves Wautelet, my promoter, for his guidance throughout this thesis and for his instruction in the program's course modules, which provided essential knowledge for this research.

I also thank Prof. Stephan Poelmans for his course modules that contributed to my understanding of the subject matter and facilitated the completion of this thesis.

I am grateful to my parents for funding my master's program and making this educational opportunity possible.

Thank you to all interview participants who contributed their time and expertise to this research: Miguel Baltazar, Dennis Cardinaels, Metin Ferati, Prof. Malgorzata Pankowska, Charlie Jessop, Mario Cunha, Oubaida Ben Yaacoub, and those who preferred to remain anonymous. Their insights were essential to this study.

Table of contents[1]

[1] This document might contain texts from earlier submitted documents within the same educational programme, related to the Master's Thesis process of the same author as the author of this work.

List of Abbreviations

Abbreviation	Full Form
AI	Artificial Intelligence
BDD	Behavior-Driven Development
CLI	Command Line Interface
ERP	Enterprise Resource Planning
EU	European Union
HR	Human Resources
IoT	Internet of Things
IT	Information Technology
LCNC	Low-Code/No-Code
LeSS	Large-scale Scrum
MBA	Master of Business Administration
PO	Product Owner
RQ	Research Question
SAFe	Scaled Agile Framework
SAP	Systems, Applications & Products
SDLC	Software Development Life Cycle
SM	Scrum Master
TDD	Test-Driven Development
UML	Unified Modeling Language
XP	eXtreme Programming

Note: This list includes abbreviations found in the main body of the thesis (Chapters 1-5), excluding those appearing only in appendices and references.

1 Introduction

Over the last few years, countries in the European Union (EU) have been facing a labor shortage of skilled digital professionals (Aksenova et al., 2024). Citizen development could address such shortages (Hoogsteen and Borgman, 2022). The creation of business applications by individuals without an IT background is commonly known in the literature as citizen development (Hoogsteen and Borgman 2022). The citizen development phenomenon can be attributed to the emergence and adoption of low-code/no-code platforms (Hoogsteen and Borgman, 2022). Low-code/no-code (LCNC) platforms enhance the firm's business agility as it mitigates developer shortages by minimizing dependencies with dedicated IT professionals (Khalajzadeh and Grundy, 2025). While shortage of skilled IT labour is a concerning issue, the use of LCNC platforms may also result in possible cost reductions as development teams are downsized and replaced with automation (Golov and Myl'nik, 2023).

The use of IT is increasingly expanding across various business functions (Promegger et al., 2021). The role of IT is increasingly becoming more strategic in businesses and organisations (Alt et al., 2020) and digital transformation of organisations will only intensify in the coming years (Alt et al., 2020). Low-code platforms will foreseeably amplify the digital transformation process of organisations in the coming years (Vincent et al., 2019). Golov and Myl'nika (2023) reported that 90 percent of time can be saved in digital transformation projects by utilising LCNC. Furthermore, LCNC platforms can reduce the barrier for innovation through user-driven creativity, thus creating value for the business (Callinan and Perry, 2024).

Combining Low-code/no-code (LCNC) approach with Agile methodologies can result in efficiency gains for businesses (Hanson, 2024). The citizen development promotes the agility of a company given limited resources (Callinan and Perry, 2024). Agile methodologies have been gaining popularity in the world of software development in the recent years (Edison et al., 2022). Agile methods focus on iterations and testing, has less of a procedural approach (Thesing and Feldmann, 2020) and is more flexible compared to other project management methods (Lutwama et al., 2024). Scrum, eXtreme Programming (XP) and Kanban are three of the most common agile frameworks (Lutwama et al., 2024). Agile methods have been found to increase customer satisfaction (Putta et al., 2018) as well as employee satisfaction (Stettina et al., 2021). While there are multiple research materials addressing the benefits of adopting LCNC development and the benefits of adopting Agile methods, little research have been done to assess the impact of LCNC development on Agile development teams. This research aims to address that gap and find the effect of LCNC development on agile development teams.

2 Literature Review

To study how LCNC development impacts agile teams, a review of relevant existing literature study and related works is carried out. This section addresses first the LCNC development followed by Agile Methods before finally addressing the relationship of LCNC and Agile.

2.1 Low-code/No-code platforms

Low-code applications can be used in different business purposes in different industry settings (Phalake et al.,2023). LCNC development makes business process automation easier and less complex (Luo et al., 2021). Picek (2023) illustrated several use cases of applications made with LCNC development in the field of enterprise resource management (ERP). Zielinski (2021) wrote about how employees of a company's human resource (HR) department utilised LCNC development to build an application that streamlines HR-related work. In China, a growing number of companies are adopting the LCNC approach to develop enterprise resource management (ERP) applications (Tang, 2021). Kalaivani et al. (2024) utilised both LCNC development and Agile methods to develop a visitor management application for a hospital. LCNC development can make it easier for people to create applications for Internet of Things (IoT) devices (Chen et al., 2022). From an organisational perspective, LCNC is bound to change the roles and responsibilities of employees, organisational culture (CIO, 2023), and decision-making processes (Bock and Frank, 2021).

As of 2020, there are over 200 Low-Code/No-Code platforms available in the market (Sahay et al., 2020). Microsoft Power Apps, Mendix and OutSystems are the current leaders of LCNC development platforms (Matvitskyy et al., 2024). Low code platforms are defined as platforms designed to facilitate fast application development, deployment, execution, and management (Vincent et al., 2020). Low code/no code platforms is characterised by its model-driven or visual development approach without the need for prior computer programming knowledge (Ang, 2021). This means users can easily build digital products by dragging and dropping components with little to no need of coding (Woo, 2020). The adoption of low-code/no-code platforms by companies is driven by enhanced software development efficiency, lower barriers to application creation and faster time to market (Käss et al., 2022). Developer shortages is also one of the factors of LCNC adoption (Guthardt et. al, 2024). Businesses can save on development costs as lesser resources are being used when they adopt LCNC solutions to develop a product (Jaglan and Upadhay, 2022). LCNC platforms have the potential to substantially shorten the software

development lifecycle (SDLC) (Bhattacharyya and Kumar, 2023) therefore minimising costs (Sanchis et al., 2020; Rokis and Kirikova, 2023).

Applications made with LCNC platforms can perform as well as those made traditionally with code (Guthardt et al., 20) and are easier to maintain (Rokis and Kirikova, 2023). Java and Javascript are two of the most common language used in LCNC development (Luo et al., 2021). LCNC platforms have made it easier for applications to scale and for developers to fix bugs found on the application product (Sahay et al., 2020). However, it is reported that LCNC platforms that are used to build the application itself can be buggy at times especially when designing applications (Liu et al., 2024). Applications developed by LCNC platforms may also have security vulnerabilities and performance issues (Woo, 2020; Ramesh and Divya, 2024; Hurlburt, 2021). The lack of proper documentation processes is one of the reasons why businesses avoid building applications with LCNC approach (Käss et al., 2023). LCNC platforms lack interoperability as standards and conventions differ depending on the platform developer, hence leading to vendor lock-in for end users of the platforms (Sahay et al., 2020). In addition to that, privacy concerns regarding data and intellectual property rights are also possible areas of concerns for end users of LCNC platforms (Wolff, 2019). While LCNC development reduces complexity in building web and mobile applications, there still exist a relatively steep learning curve in using LCNC platforms (Luo et al., 2021). Kandaurova (2024) found that LCNC platform users still need to have a basic knowledge in code to make a functioning application.

2.2 AI and LCNC

The current research landscape regarding the use of AI in LCNC development is currently limited while there is growing interest among businesses (Kandaurova, 2024). Incorporating AI into LCNC platforms enhances business agility by enabling real-time, data-driven decision-making (Kok et al., 2024). LCNC platform vendors have started integrating Artificial Intelligence (AI) features that assists builders in making their applications (Woo, 2020). The use of AI prompt engineering to model web and mobile applications can result in LCNC platforms into “true no-code platforms’ allowing for faster iterations for Agile teams (Hagel et al., 2024). ChatGPT’s large language model can generate boilerplate code templates to build applications (Martins et al., 2023). Integrating AI and LCNC approach can ensure adherence to standardization and best practices (Martins et al., 2023).

2.3 LCNC impact on Technical Debt

Schmidt's (2016) found that software quality is a determinant of the performance of Agile development teams. This can be be related to the concept of technical debt. Technical debt refers to the backlog of work that development teams accumulate on a product when they take shortcuts during the development process prioritising speed for product delivery (Pavlic et al., 2022). It is clear how the LCNC approach makes it easier for applications and digital products to be developed. While LCNC development can reduce technical debt as there are less poorly written code when building applications (CIO, 2023), Havelund and Steffen (2021) along with Lethbridge (2021) argued that LCNC approach will lead to more technical debt as the LCNC approach lacks functionalities that facilitate the standard industry processes for software engineering such as version control. LCNC development can result in code that are too complicated which adversely impacts technical debt (Lethbridge, 2021; Havelund and Steffen, 2021) as it is harder to document and be reutilised for future use (Lethbridge, 2021). From a management perspective, technical debt can be reduced if enterprise architects simultaneously assume the role of product owner in agile settings (CIO, 2024). It is still not clear yet how the citizen developer trend that companies are adopting have an impact on technical debt (Barkin and Davenport, 2023).

2.4 Agile Methodologies

The agile methodology which was designed for software development is very much related to the principles and concept of lean that was originally meant for manufacturing processes (Babik, 2018). Agile Development is one of the most favoured methodologies of software development in the twenty-first century because of its adaptability in fast-changing environments where requirements change rapidly (Al-zewari et al, 2017). Agile methodologies emphasize iterative development and continuous improvements, taking a less rigid, procedural approach (Thesing & Feldmann, 2020) and offering greater flexibility than traditional project management methods such as Waterfall (Lutwama et al., 2024).

Agile methodologies are widely adopted in the tech industry because it has been proven to speed up the product's time to market in dynamic environments where requirements change at a fast rate (Papatheochaorus and Andreou, 2014). In its essence, the Agile model focuses on collaborative interaction (Babik, 2018) and continuously incorporating stakeholder feedback, particularly from customers, through small but frequent product release and iterative updates, which leads to continuous evaluation and refinement (Maassen, 2018). It has been shown that using Agile methodologies positively impact an organization's overall performance especially in terms of productivity and employee

satisfaction (Stettina et al., 2021). Agile methodologies are not restricted to applications only in the tech sector (Breyter, 2022). Business units can achieve business agility by adopting agile into their workflow. Adopting agile practices improves in the business unit's capacity to respond rapidly to market shifts—staying efficient, customer-focused, and cost-effective—while still maintaining high standards of quality (Breyter, 2022).

2.5 Scrum

2.5.1 The Scrum Process

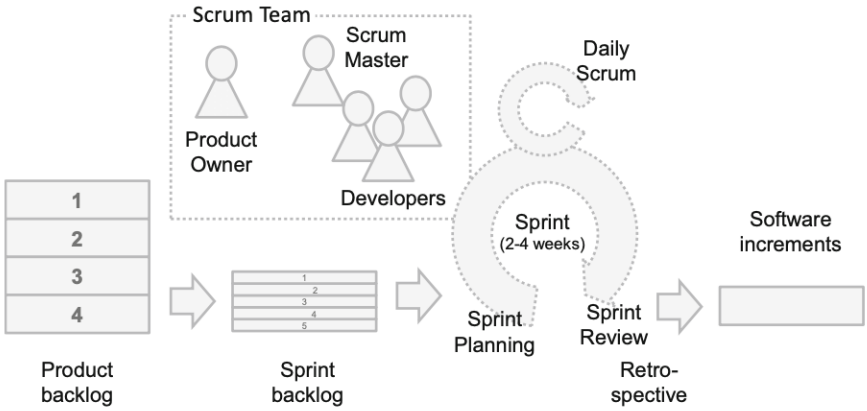
Scrum is one of the widely used frameworks for the agile methodology (Maximini, 2018). Scrum has a more dynamic, process-based, and flexible approach compared to eXtreme Programming (XP) which focuses more on coding methods (Babik, 2018). Self-organizing teams is a key feature of the agile scrum teams. Teams consists of members with a variety of skills and specializations unlike conventional teams where individuals are grouped according to their skill set (Bass, 2022).

The Product Owner and Scrum Master are the leading roles of the scrum team along with architects, developers, technical writers, and quality managers (Babik, 2018). Schmidt (2016) on the other hand suggested further that the development teams should not have other specialist roles than programmers and that these programmers should have necessary skills to make the team as cross-functional, lean, and flexible as possible.

The Product Owner (PO) acts as the customer's proxy on the team, articulating their needs, managing and organising the team's objectives for the upcoming Sprint, and ensuring that the work delivers real value to the customer (Schmidt, 2016; Kelly, 2019; Noll et al., 2017). The Scrum Master (SM) serves as a facilitator and coordinator who upholds the Scrum processes and removes any obstacles that could prevent the team from working effectively (Schmidt, 2016; Shastri et al., 2021; Noll et al., 2017). While the function and title of project manager is non-existent in Scrum, Shasri et al. (2021) proposed that the Scrum master acts and functions as the de-facto project manager in Scrum teams. The Scrum master facilitates Scrum ceremonies such as the sprint planning, sprint review and daily standups (Noll et al., 2017).

Figure 1

Scrum process diagram (Schmidt, 2016)

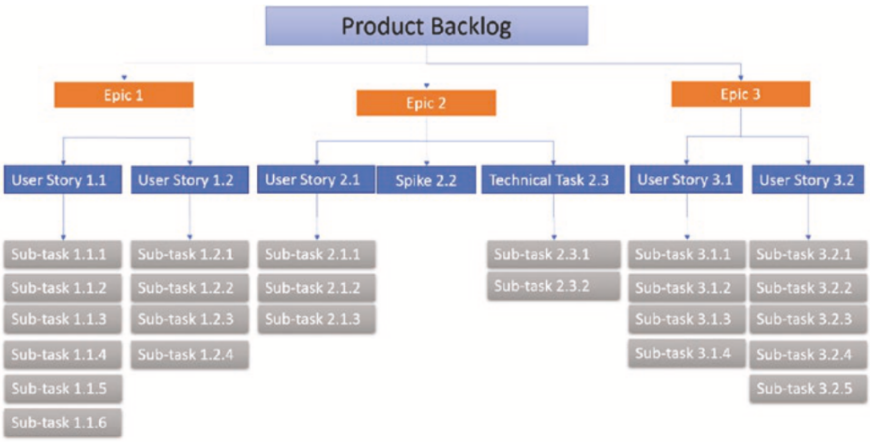


Note. Adapted from Agile software development teams: The impact of agile development on team performance (p. 17), by C. Schmidt, 2016, Springer. © 2016 Springer.

The scrum essentially method breaks down the project into more workable smaller iterative increments called sprints (Babik, 2018), in which developers are given the autonomy to plan and manage the tasks assigned prior in sprint planning (Schwaber and Sutherland, 2020). Product features are defined by the product owner in the product backlog and are called epics. Epics are further decomposed into user stories; a technology agnostic description written from the point of view of the user (Bass, 2022). User stories includes sub-tasks for the developers to work on (Breyter 2022).

Figure 2

Product Backlog hierarchy (Breyter, 2022)



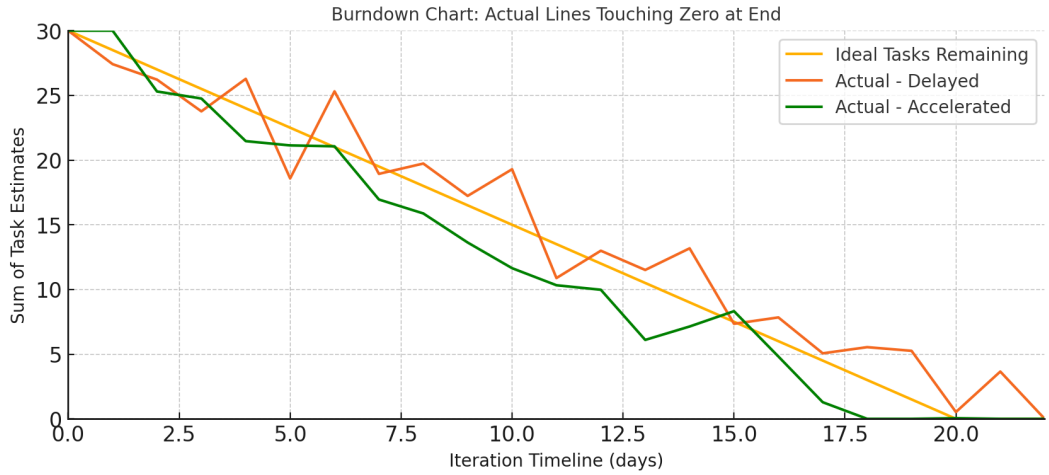
Note. Adapted from Agile product and project management: A step-by-step guide to building the right products right (p. 117), by M. Breyter, 2022, Apress. © 2022 Apress.

The work defined in the product backlog is further distributed and decomposed in the sprint backlog for each sprint iteration, which lasts for two to four weeks (Tal, 2015). Before each sprint, a sprint planning meeting is organised to organise the deliverables for the upcoming sprint (Tal, 2015). During sprint planning, the team selects which product-backlog stories can be completed in the upcoming sprint by considering their velocity, available capacity, and the estimated effort for each story (Noll et al., 2017). Kanban is a commonly utilised tool to track and manage the work items and user stories (Breyter, 2022). Daily meetings facilitated and coordinated by the scrum master are held during each sprint to update each other daily on the work done (Babaian, 2019). At the end of each sprint, the team convenes once again for a Sprint retrospective meeting where performance is reviewed (Tal, 2015). s and bugs found after each sprint are listed as working items to be worked on again in upcoming sprints by team members responsible for quality assurance (Tal, 2015).

2.5.2 Measuring Performance in Scrum teams

Figure 3

Hypothetical Burndown chart of an Agile Project (AI-Generated, 2025)

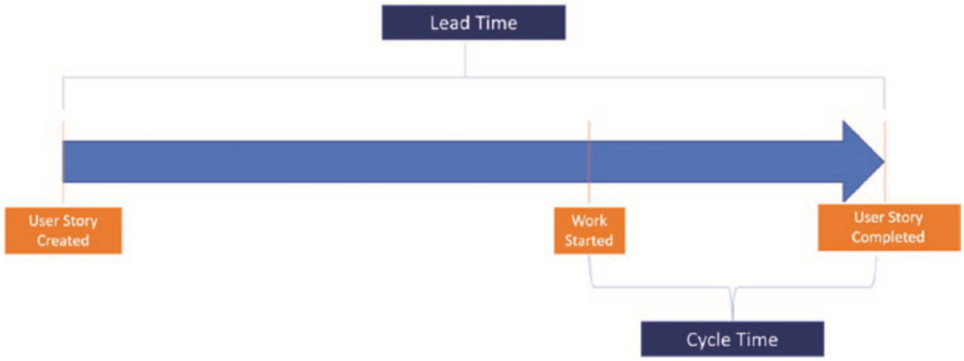


Team performance is perceived to be the most important metrics in an agile project and velocity is one of the determining factors (Almeida & Carneiro, 2023). However, velocity as a metric is not very reliable unless combined with cycle and lead time metrics (Almeida & Carneiro, 2023). In each Scrum project, a burndown chart is used to measure the progress of work done (Tal, 2015). The x-axis shows the time in calendar days and the y-axis shows the amount of work done. A negative-sloping straight linear line called the burndown line is drawn as a reference for the ideal work velocity, with flat lines representing weekends or days where no work is expected to be done (Breyter, 2022; Tal, 2015). The actual progress line is drawn as the days pass, by subtracting the amount of work done, and is not necessarily linear (Breyter, 2022). If the team completes tasks faster than expected, then

the actual progress line would appear lower than the reference line and if the team completes tasks slower than expected, the progress line would appear higher than the reference line. This is shown in Figure 3. In addition to the burndown chart that measures progress in terms of work velocity, lead time and cycle time is measured (Breyter, 2022). When using Kanban, lead time is the time measured from the inception of a user story till its completion whereas the cycle time is the time measured from the point the user story is worked on till its completion. Both lead time and cycle time of user stories can be averaged out to serve as additional metrics for team performance.

Figure 4

Lead & cycle times (Breyter, 2022)



Note. Adapted from Agile product and project management: A step-by-step guide to building the right products right (p. 199), by M. Breyter, 2022, Apress. © 2022 Apress.

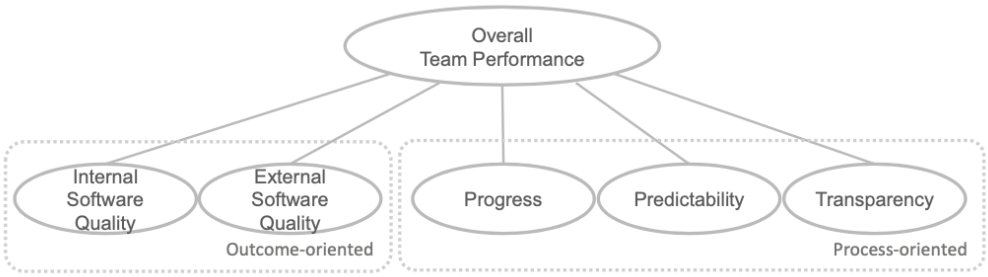
Schmidt (2016) investigated the impact of Agile on development teams in SAP SE with empirical methods. In his research that was compiled into a book, team performance was broken down into several dimensions, namely: internal and external software quality, which is categorised as outcome-oriented performance, in addition to progress, predictability, and transparency, which are categorised as process-oriented performance (Schmidt, 2016). Of these dimensions, progress, software quality and transparency are statistically significant (Schmidt, 2016). The progress dimension is determined by velocity and speed of work. The transparency dimension is determined by external and internal communication of teams (Schmidt, 2016). Software quality is determined by code reusability and maintainability, stakeholder satisfaction, and the conformity and compliance of the software to the client's requirements and requested functionalities (Schmidt, 2016).

Software maintenance is formally defined as making changes to software after it has been delivered, with the goal of fixing bugs, enhancing performance or other features, or adjusting the software to work in changed conditions. Maintainability describes how easily

and efficiently these maintenance tasks can be performed (Jain et al., 2018). Lopez et al (2022) further emphasised that maintainability, reliability and efficiency are important factors that contribute to software quality in the context of agile and rapid development.

Figure 5

Schmidt (2016) proposed dimensions of performance.



Note. Adapted from Agile software development teams: The impact of agile development on team performance (p. 93), by C. Schmidt, 2016, Springer. © 2016 Springer.

2.4.3 Evolution of scrum

Scrum serves as a base for many of the scaled agile frameworks as shown in Table 1 (Breyter, 2022). The Scrum method was originally designed for smaller teams and has gone through evolutions for larger scale applications in organisations and projects resulting in new frameworks such as the Large-scale Scrum (LeSS), the Scaled Agile Framework (SAFe) and the Spotify Method (Uludag et al., 2021). The concept of ‘scrum of scrums’ is a key feature of LeSS, in which a team consisting of the area scrum master's and area product owners of individual scrum teams working on a larger goal or product is formed (Bass, 2022). In addition to that, new roles and teams are introduced to assist with the coordination between the business and technical sides such as the Governor and Product Manager for LeSS (Bass, 2022), and the Release Train Engineer and Portfolio teams for SAFe (Putta et al., 2018).

Table 1

Scaled Agile Frameworks (Breyter, 2022)

	Type and size of the organization	Methods and practices adopted	Specifies portfolio management and business prioritization	Establishes a collaborative dependency management mechanism	Professional association that endorsed the framework
Scaled Agile Framework (SAFe)	50–124 people in an Agile Release Train, scaling indefinitely	Scrum, Kanban, Lean, DevOps, XP Practices	In detail: Lean portfolio management and business prioritization based on Value Stream Mapping (VSM)	Advanced dependency management starting with long-term planning	Scaled Agile Academy
Large-Scale Scrum (LeSS)	LeSS Huge allowed for thousands of people	Scrum primarily	By scaling product management	Via structured ongoing collaboration	LeSS practitioner groups
Disciplined Agile Delivery (DAD)	200 people or more	Full toolbox: Scrum, Agile modelling, Unified Process, XP, TDD, Agile	Focuses on technical practices	Via collaboration	Project Management Institute
Spotify Scaling Model	Around 300 people	Allows the team to choose any Agile framework	Via Tribe-specific accountability and the role of product management	Organic	N/A
Scrum@Scale	No limitations imposed; successes limited to several hundred	Scrum	Product Owner collaboration	Via Scrum of Scrums	Agile Alliance
Nexus	Three to nine Scrum teams	Scrum	Product Owner collaboration	Advanced dependency management via Integration Team	Scrum Alliance

Note. Adapted from Agile product and project management: A step-by-step guide to building the right products right (p. 255 – p. 256), by M. Breyter, 2022, Apress. © 2022 Apress.

2.6 Implications of LCNC platforms on Agile methods

Rokis and Kirikova (2022) found that the principles of agile methods are compatible with the essence of LCNC development. Integrating low-code/no-code (LCNC) with Agile methodologies enhances efficiency and adaptability for businesses (Hanson, 2024). Combining Low-code/no-code development (LCNC) and Agile methodologies like Scrum can significantly accelerate application development while enhancing an organization's efficiency, quality, and speed, thus leading to success (Abdul Razak et al., 2024). In a study by Lebens and Finnegan (2021), LCNC platforms is proven to be a useful tool to teach university students agile methods. Kalaivani et al. (2024) found that Agile methods are

useful as a project management tool when developing a healthcare application with a low-code/no-code platform. The roles in agile team would be even more dynamic and cross-functional with LCNC development (Elshan et al., 2024). Product owners must adapt to the faster software development cycles (Elshan et al., 2024). To study the impact of LCNC development on Agile methods, it is useful to draw parallels between the findings of existing literature with some of the twelve principles of Agile.

Maintaining simplicity by reducing complexities in both processes and the actual software product itself is a key principle of Agile Manifesto (Agile Alliance, n.d). LCNC platforms simplify the complex process of conventional coding (Käss et al., 2022), enabling users to streamline workflows and accelerate the software development process (Picek, 2023) Collaboration between businesspeople and developers is another driving principle behind the Agile Manifesto (Agile Alliance n.d). The use of LCNC platforms in developing software applications promotes collaboration between IT and business teams (Tang, 2022), streamlining Agile development workflows (Appian Corporation, 2019). Making software development more accessible through LCNC platforms fosters collaboration across departments, driving innovation by reducing the burden on dedicated IT teams and thus accelerating the development process (Ramesh and Divya, 2024). LCNC platforms empower business users to create applications with lesser dependence from dedicated IT specialists, thus minimising workload (Käss et al., 2023; Matook et al., 2025). In short, LCNC development unites business and technology as it aligns with the principles of agile (Tang, 2021).

Keeping customers satisfied by incrementally delivering useable software products early and consistently with speed and frequency are the principles behind Agile Manifesto (Agile Alliance, n.d). Low-code/no-code platforms align well with Agile methodology, as iterative development and frequent delivery is prioritized (Razak et al., 2024). Low-code development enables rapid prototyping and development of products, allowing developers to assess customer needs and validate ideas before allocating resources to product features that may hold little value (Sanchis et al, 2020). Being flexible with changing requirements from customers is one of the principles of Agile (Agile Alliance n.d). Low-code platforms enable developers to adapt to changing requirements as they are able modify applications in response to evolving processes and requirements, incorporating user feedback for continuous improvement faster (Wolff, 2019). Since developers can work rapidly on iterations based on feedback, this can reduce requirement inconsistencies (Alamin et al.,2023). The LCNC development approach is more agile compared to conventional approach as developers are dealing less with complicated code (Shridhar and Bose, 2021). Hence adopting the LCNC approach will improve customer satisfaction (Rokis and Kirikova, 2022).

However, it has also been reported that LCNC development may pose some challenges in Agile teams. Conflicts between business users and IT teams over control and governance may happen in LCNC development, thus hindering Agile collaboration (Khalajzadeh and Grundy, 2025). Utilizing low-code and no-code tools could result in reduced control over security and governance (Waqas et al., 2024). Thus, a unified IT strategy and governance framework is essential to resolve such issues (Ajimati et al., 2025).

3 Research Methodology

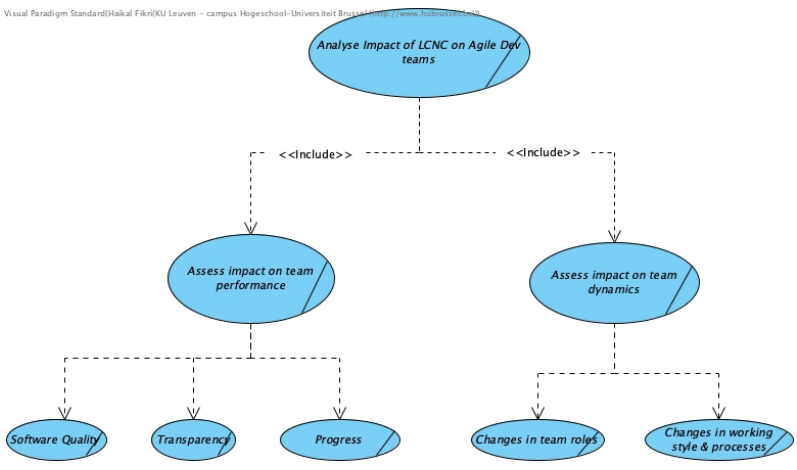
While there has been multiple research that focuses on LCNC development and Agile methods individually, there has been limited studies on how both concepts impact and interact with each other. This qualitative study aims to address that gap by studying the impact of LCNC approaches on agile development teams through interviews with practitioners and relevant industry experts. The following research question formulated below must be answered to study the organisational impact of LCNC on agile development teams:

RQ1: How does the use of Low-code/No-code platforms impact Agile development teams?

To analyse how the use of LCNC platforms impact Agile development teams, performance and team dynamics are two overarching factors that needs to be investigated. To assess the impact of LCNC on team performance, this research will use only the statistically significant dimensions of Agile team performance that was proposed by Schmidt (2016). Therefore, software quality, progress and transparency are used to assess the impact of LCNC on team performance. To assess the impact of LCNC on team dynamics, the changes in team roles, working style, and working processes resulted by the use of LCNC platforms will be investigated. Of all the existing Agile frameworks, only Scrum will be considered in this research as it is the most prevalent and widely used framework in practice. Such consideration is made because Scrum is the basis for many of the scaled frameworks as reported by Breyter (2022).

Figure 6

UML use-case goal diagram of research



3.1 Interview Design

The interviews follow a realist ontology with a neo-positivism epistemology approach with the following research objectives that will answer the research question:

1. Investigate the effect of LCNC approach on team performance with software quality, progress, and transparency as metrics.
2. Investigate the possible changes in team dynamics in terms of roles, working style, and working processes caused by the adaptation of LCNC approach.

The interviews are semi-structured and exploratory with questions that covers the two overarching factors that determine the impact of LCNC on Agile development teams. Respondents are selected through non-probability sampling, specifically with purposive sampling based on relevance to the research topic. Interviewees are chosen through personal network connections and LinkedIn based on two main criterions. Firstly, their professional experience in the area of LCNC. This can be employees of LCNC platform vendors and external consultants or persons who are users of the platforms. Secondly, their familiarity with Agile frameworks. Ideally, the interviewee should have a professional Agile certification, however, the minimum is that they should be at least exposed to the Agile working environment sometime in their career. Appendix 1 lists the relevant background information of all interviewees. Table 2 below describes the interview questions related to background that are asked in the beginning followed by table 3 where the questions pertaining to the dimensions that the overarching factors depend on are listed.

Table 2

Introduction Questions

No.	Question
1	Please introduce yourself.
2	Briefly explain your working experience.
3	Which agile framework are you most familiar with?
4	Which LCNC platform are you familiar with?

Table 3

Interview Questions

Factor	Dimension	Question
Performance	Software Quality	How does LCNC affect the team's defect rate per sprint and what evidence links any observed increase or decrease directly to the LCNC implementation?
	Software Quality	On a scale of 1 (very low) to 7 (very high), how would you rate the maintainability of LCNC deliverables compared to traditional code? Please explain the rationale behind your rating with a recent example.
Performance	Transparency (Internal Communication)	On 1 (very difficult) to 7 (very easy), how smooth is cross-member coordination when working with LCNC platforms? Can you illustrate a recent example?
	Transparency (External Communication)	On 1 (slow/infrequent) to 7 (rapid/frequent), how would you rate the pace of stakeholder feedback when delivering work made with LCNC development? Please explain the rationale behind your rating.
Performance	Progress	How has adopting LCNC shifted your team's actual burndown line relative to the ideal trajectory—indicating overall acceleration or deceleration of work—and what project-level observations support that pattern?
	Progress	How can the adoption of LCNC impact the cycle time and lead time? Can you provide a reason or an example behind your answer?
Team Dynamics	Changes in Roles	How has adopting LCNC influenced the composition and number of roles on development teams—specifically, which new LCNC-oriented roles have emerged, and which traditional scrum/agile roles have diminished—and can you illustrate these shifts with concrete examples and use cases?
	Changes in Roles	How has adopting LCNC shifted the hands-on responsibilities of product owners—specifically, have they taken on more configuration or design work—and can you illustrate that change with a concrete use-case or example?
Team Dynamics	Changes in working style	How has LCNC adoption affected the coordination and facilitation duties of the Scrum master, and can you share a real project scenario that demonstrates the change?
	Changes in working style	How has LCNC adoption changed your backlog prioritisation process, and can you share a specific example where LCNC led you to reprioritise an item?
Team Dynamics	Impact of Gen AI	How would the emergence of generative AI impact LCNC approach and Agile altogether?

3.2 Interview Participants Selection

Table 4 below lists the eight interview participants who agreed to participate in this research. A total of 80 people were approached on LinkedIn Sales Navigator. For this research multiple search queries with the terms “Agile”, “Scrum” and the respective and relevant LCNC platforms were executed. Prospects are screened and selected carefully based on the criterion in § 3.1 by analysing their job position and experience, certifications and skills listed on the individual profile page. An invitation message was then sent to prospects who pass the screening process. In addition to that, personal connections and referrals were used to approach some of the participants. In the end, only nine prospects responded positively, of which only eight managed to successfully do the interview. Only one out of the eight respondents chose to remain anonymous. All eight interview participants indicated that scrum is the main methodology that they are most familiar with.

Table 4

Interview Participants

Name	Job Experience	LCNC Platform Experience
Miguel Baltazar	VP Developers, OutSystems	OutSystems
Dennis Cardinaels	Tech Lead Capgemini Belgium	Outsystems
Metin Ferati	Founder, N’Katrori & Freelance Senior developer	Wordpress, Replit, V0 by Vercel, Cursor
Prof. Malgorzata Pankowska	Professor, University of Economics Katowice	WebCon
Charlie Jessop	Solution Architect, Capgemini Belgium	Outsystems
Anonymised participant	Software Developer in Latvia	Microsoft Power Apps
Oubaida Ben Yaacoub	Freelance Business Analyst/Product Owner	Microsoft Power Apps
Mario Cunha	Tech Lead, OutSystems	Outsystems, Mendix

3.3 Data Collection and Analysis

The interview data collection was done fully online through Microsoft Teams over a period of one month between July and August 2025. All participants have given their consent for the interview to be recorded and transcribed. After transcription is done, thematic coding analysis was performed. Table 4 shows the main themes that was identified from all the interviews and description of the Appendix 1 to Appendix 8 is the transcripts for the eight

interviewees above. The full excel sheet that contains the main themes, child themes and excerpts can be accessed in the drive link in Appendix 9.

Table 5

Thematic Codes

Theme	Description
Software Quality & Maintainability	Covers defect rates, error types, governance issues, platform limitations, and ease/difficulty of maintaining LCNC deliverables.
Communication & Collaboration	Covers cross-member communication, stakeholder feedback loops, citizen developer interactions.
Delivery Speed & Workflow Efficiency	Covers delivery pace, quick wins, and changes in backlog prioritization due to LCNC.
Role & Responsibility Changes	Covers shifts in Product Owner responsibilities, Scrum Master, developer roles, hybrid positions and responsibilities.
AI Integration in LCNC	Covers use of AI tools in LCNC development, AI-assisted coding, Automation, perceived future impact.

3.3.1 Building a Transcriber Web App with LCNC

A custom web-based transcription application was developed using a combination of low-code, and no-code tools to facilitate the transcription and analysis of qualitative interview data for this study. The application was designed to transcribe the eight interviews conducted with participants. The front end was made entirely through prompting with Anthropic's Claude Code command line interface (CLI) that was installed locally running Claude AI's Opus 4 large language model (LLM). The backend was developed and deployed using n8n, a low-code workflow automation platform, enabling seamless integration between the application's frontend, OpenAI's GPT-4 transcription LLM and Google Docs.

AI-powered transcription was employed to reduce the time and manual effort required for converting audio data into text while maintaining a high degree of accuracy. The application was created to improve efficiency in processing large volumes of qualitative data and to demonstrate the applicability of low-code/no-code platforms in practical research scenarios, consistent with the study's focus on their impact within Agile development environments.

3.3.1.1 Build Process of Transcriber Web App

The application was designed with a workflow in which the frontend enables the upload of interview recordings through a dedicated form. The form graphical user interface (GUI) was coded in HTML with CSS styles applied and AJAX enabled. Upon upload and submission, the data from the upload form is captured via a webhook, which initiates the backend processing. This data is transmitted to an n8n development server, where it is routed to the OpenAI integration node through an API connection. The integration with OpenAI's GPT-4 transcription model facilitates the automated transcription of the uploaded audio or video files albeit with an upload limit of 25 Megabytes per file. As a result of that limitation, the interview videos were converted to a low bitrate .mp3 file with Adobe Audition. Following the Open AI transcription, a respond-webhook node is triggered to confirm the successful execution of the process. Simultaneously, another node within the workflow appends the transcribed text to a designated Google Docs file, ensuring that all interview data is consolidated in a single document for analysis. This was made possible by connecting Google's API key with n8n.

3.3.1.2 Using the Transcriber Web App

This workflow was iteratively applied for all eight interviews, with each transcription appended sequentially to the same document on Google Docs. The entire process was configured and optimized within the low-code environment of n8n, leveraging its modular node-based automation capabilities. Current development efforts are focused on expanding the application's functionality to include the ability to export transcriptions directly as Microsoft Word documents, thereby enhancing its usability for qualitative research documentation and deploying the app with Railway. Future development efforts could include the addition of another AI node which would be able to re-check and edit the transcription for mistakes through context and even generate a short summary of transcripts.

Figure 7 shows the n8n workflow. Figure 8 and 9 show the simple HTML frontend coded by Claude Opus 4 model. The time taken to build the prototype of the app was roughly about 3 hours with significant amount of time taken to familiarise with n8n, given no prior experience with the platform, and to troubleshoot API connectivity issues of nodes. OpenAI's Chatgpt 5 was additionally utilised to consult when troubleshooting issues. The development cost incurred specifically for this prototype app in terms of API usage and platform usage totalled up to less than 5 dollars, excluding the overhead costs of n8n and Claude AI licences.

Figure 7

n8n workflow

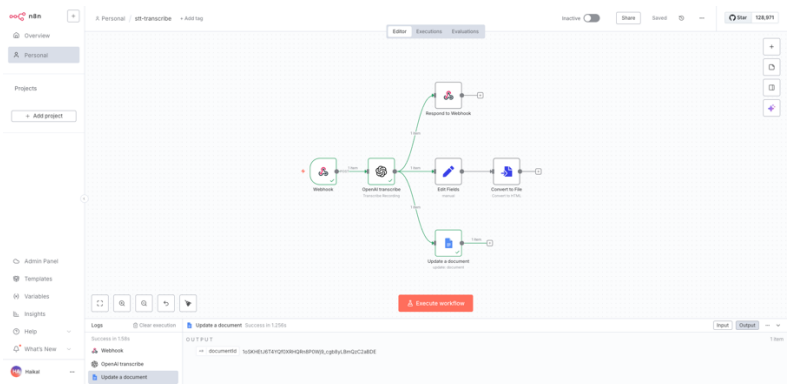


Figure 8

Frontend GUI form

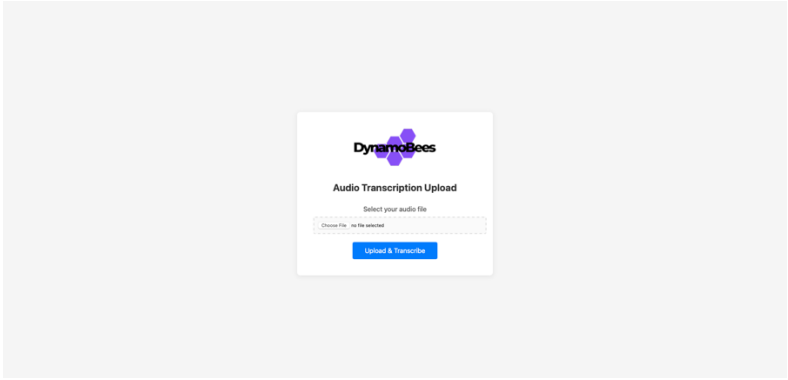
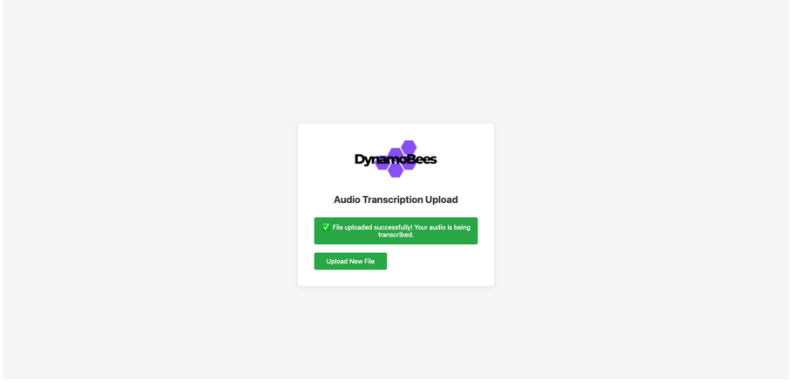


Figure 9

Frontend GUI form upon submission



4 Discussion

In this section, each thematic area listed in Table 5 identified through the analysis of interview data will be examined in detail, comparing the empirical findings with existing literature and theoretical frameworks. The analysis of interview data revealed five main thematic areas that characterize the impact of LCNC on Agile teams: software quality and maintainability, communication and collaboration, delivery speed and workflow efficiency, role and responsibility changes, and the emerging influence of AI integration. These themes align with and extend Schmidt's (2016) performance dimensions framework, while revealing nuanced effects that vary significantly based on project complexity, team maturity, and organizational context.

4.1 Software Quality & Maintainability

Defects to a software can be used to determine the quality of the end-product. Defects can appear because of developer error or in the case of LCNC approach, because of the platform. Defects essentially creates more work for the Agile team and its accumulation in the subsequent Sprint backlog could lead eventually delays in the overall project, which will reflect in a burndown line going above the ideal trajectory. Based on the interview data analysis, it can be concluded that LCNC approach will result in a lower defect rate to a limited extent. Defects related to frontend and syntax can decrease because of LCNC (Ferati, personal communication, 2025; Anonymous Participant, personal communication, 2025; Baltazar, personal communication, 2025), however the defect rate is very much still highly dependent on the skill level of developer and/or developing team themselves and the platform itself (Cunha, personal communication, 2025; Anonymous Participant, personal communication, 2025; Ferati, personal communication, 2025; Jessop, personal communication, 2025). Rectifying or fixing a defect can be much faster with LCNC compared to traditional development approach (Jessop, 2025; Anonymous Participant, 2025; Cunha, 2025).

Maintainability among other things define software quality (Kanellopoulos & Yu, 2015; Lopez et al., 2022). In this research, interviewees were asked to rate the maintainability from one to seven with one being low and seven being high. The term maintainability must not be conflated with maintenance. Interviewees were made clear beforehand that high maintainability in this research context means that the end software is easier to maintain (vis-à-vis low maintenance) with LCNC approach. Majority of the respondents indicated that software made with LCNC is generally easier to maintain, thus high maintainability.

However, vendor lock in issue and platform limitations may be a hindering issue when it comes to maintainability especially with more customised software (Anonymous Participant, personal communication, 2025). High overhead costs may also affect maintainability from a financial standpoint (Jessop, personal communication, 2025).

4.2 Communication & Collaboration

Transparency, as reported by Schmidt (2016), relies on internal and external communication, and affects the team performance. Majority of the respondents indicated that LCNC has a positive effect in external communications with stakeholders as they can deliver iterations at a faster speed with LCNC. This can also be attributed to the citizen development phenomenon where non-technical stakeholders are able to contribute to the development process (Anonymous Participant, personal communication, 2025).

In terms of internal communications between cross members however, respondents have mixed perceptions. Respondents with a strong background in traditional approach (Ferati, personal communication, 2025; Jessop, personal communication, 2025) gave lower ratings. The lack of version control features in LCNC platforms makes internal communication worse than traditional approaches (Jessop, 2025) and that the inclusion of citizen developers may affect internal communication negatively if citizen developers lack technical skills (Ferati, 2025). Respondents also shared that teams tend to be a lot smaller in LCNC projects, therefore streamlining internal communications (Ben Yaacoub, personal communication, 2025; Baltazar, personal communication, 2025).

4.3 Delivery Speed & Workflow Efficiency

Previous literatures have stated that combining LCNC with agile will accelerate project delivery (Razak et al., 2024; Picek, 2023) as developers are able to work more efficiently (Sanchis et al, 2020). As demonstrated in § 3.3.1 and stated by a respondent (Ben Yaacoub, personal communication, 2025), developers using LCNC approaches can deliver a minimum viable product faster as LCNC solutions make it easier for them to develop applications. While most respondents agree that teams can work faster and deliver iterations at a rapid rate with LCNC approaches compared to traditional approaches reflected through the positive response on lead and cycle times, many have mixed reactions when asked about the overall progress and time taken to complete and deliver the entire agile project.

Overall progress can be measured by the burndown line - a line that is below the ideal trajectory means an acceleration of work, whereby one that is above means a deceleration of work. Respondents indicated that there are other variables in play that will affect the acceleration and deceleration of the project delivery. With LCNC approaches, developers can work more efficient at a faster rate (Cardinaels, personal communication, 2025; Cunha, personal communication, 2025; Baltazar, personal communication, 2025; Jessop, personal communication, 2025), however bottlenecks may appear at other stages of the project and requirements may change, thus, slowing down progress (Cardinaels, personal communication, 2025; Baltazar, personal communication, 2025). The accelerating factor attributed to the use of LCNC may also be diminished when the project is more complex (Jessop, personal communication, 2025; Cunha, personal communication, 2025; Pankowska, personal communication, 2025).

4.4 Role & Responsibility Changes

Self-organising and cross functional team roles are one of the features of Agile development teams (Bass, 2022). LCNC platforms have given rise to the citizen development phenomenon, where people with none or limited technical knowledge are able to build digital products (Hoogsteen and Borgman, 2022). The opposite could be true as well - with LCNC, professional developers can now also become business owners or be more involved in other business processes (Ferati, personal communication, 2025; Baltazar, personal communication, 2025) because of efficiency gains (Cunha, personal communication, 2025).

The work responsibilities of UI/UX designers can be absorbed by developers (Jessop, personal communication, 2025) and designers can become citizen developers with LCNC (Ferati, personal communication, 2025). In addition to that, a respondent observed that there are lesser product testers in agile development teams adopting LCNC (Jessop, personal communication, 2025). Teams are observed to be smaller as team members often take multiple roles (Ben Yaacoub, personal communication, 2025; Baltazar, personal communication, 2025; Cardinaels, personal communication, 2025). However, the combining of roles may also lead to communication issues and that citizen development may not be appropriate for bigger enterprise projects (Cunha, personal communication, 2025).

The role of product owners could get more technical as LCNC is less complicated than traditional software development approaches (Ferati, personal communication, 2025; Ben Yaacoub, personal communication, 2025). LCNC approach makes it easier for product

owner to organise their backlog and user stories (Ben Yaacoub, personal communication, 2025; Anonymous Participant personal communication, 2025). Scrum masters could be able to take on more than one project and have multiple development teams with LCNC as their work becomes easier with LCNC especially with certain LCNC platforms such as OutSystems which are designed with Agile in mind (Jessop, personal communication, 2025; Ben Yaacoub, personal communication, 2025; Baltazar, personal communication, 2025; Cunha, personal communication, 2025). With LCNC and Scrum, the role of business analyst can be combined with either the product owner or scrum master (Ben Yaacoub, personal communication, 2025; Cardinaels, personal communication, 2025). With roles being combined, organisations could possibly be saving costs on human resources. This finding aligns with Khalajzadeh and Grundy (2025) proposition where low-code/no-code (LCNC) solutions enhance the firm's business agility as it mitigates developer shortages by minimizing dependencies with dedicated IT professionals.

4.5 AI Integration in LCNC

With the emergence of generative AI, it makes it even easier for one to build digital products. The interviews conducted for this thesis tries to explore how AI plays a role in the impact of LCNC on scrum given the limited research resources regarding the use of AI in LCNC development (Kandaurova, 2024). AI models such as the one used in § 3.3.1 which can be installed locally and operated with CLI are able to generate a software practically with almost zero coding knowledge. Not only it is able to build and develop something from scratch but is able to debug and troubleshoot itself. The models can code and develop software faster and may pose as a competition to LCNC vendors if not adopted (Jessop, personal communication, 2025).

LCNC platforms should be the enabler of AI-use by making it easier for developers to integrate AI agents in the application for end users (Baltazar, personal communication, 2025). As shown in the transcriber web app build in § 3.3.1 and mentioned by Ferati (personal connection, 2025), it is possible to have AI powered automation with LCNC approaches. AI would be a good addition to LCNC as a copilot and as a decision support assistant, thus making the work of the developer even easier, increasing their productivity and accelerating the delivery even more than the current rate with LCNC solutions (Ferati, personal communication, 2025; Anonymous Participant, personal communication, 2025; Cunha, personal communication, 2025; Pankowska, personal communication, 2025; Baltazar, personal communication, 2025).

5 Conclusion

This research set out to answer the fundamental question:

How does the use of Low-Code/No-Code platforms impact Agile development teams?

By adapting Schmidt's (2016) agile performance dimensions framework to LCNC contexts, the study validates the continued relevance of software quality, transparency, and progress as key performance indicators of agile teams while revealing how these dimensions manifest differently in LCNC environments.

Through comprehensive analysis of interview data from eight practitioners across various LCNC platforms and organizational contexts, several key insights have emerged that contribute to the current research landscape of LCNC, citizen development and agile. LCNC platforms demonstrate a generally positive impact on software quality through reduced frontend defects and improved maintainability, though this benefit is depends on management processes, business context and complexity of project.

With regards to transparency, communication and collaboration patterns reveal a divergent impact between internal and external stakeholder interactions. External communication with stakeholders may be significantly improved. This improvement is attributed to faster iteration delivery and the ability for non-technical stakeholders to participate directly in the development process through citizen development. However, internal team communication presents a more conflicting picture. While smaller team sizes can enhance communication efficiency, LCNC platform limitations such as limited version control capabilities and platform specific features can create new communication barriers.

With LCNC, there is bound to be a fundamental restructuring of traditional Agile roles. Teams operating with LCNC platforms tend to be significantly smaller, with members assuming hybrid responsibilities. The consolidation of traditional roles like UI/UX designers into developer responsibilities, and business analysts' roles with scrum master's or product owners' responsibilities because of LCNC adoption represents a shift toward even more cross-functional team structures that align with Agile principles.

While LCNC platforms can accelerate development and improve workflow especially for simpler applications, this advantage diminishes with project complexity. As Miguel Baltazar noted, the removal of development bottlenecks through LCNC causes bottlenecks to shift

to other areas in a project thus necessitating a holistic adoption of Agile methods rather than simply adopting LCNC tools. This research provides initial evidence that LCNC platforms, when properly implemented within appropriate contexts, can enhance Agile team performance and enable new forms of cross-functional collaboration.

With generative AI, it is even easier to develop applications with the LCNC approach as shown in § 3.3.1, thus making the job of the developer easier. This could potentially result in developers being more efficient and productive. By combining AI and LCNC approaches with Agile, the delivery of a digital product could be accelerated even further. In addition to that, software made with LCNC have the possibility of AI agents enabling automation of certain processes.

While this study tries to be as representative as possible by interviewing developers, LCNC vendors, academics and business analysts, this study's limitations include its reliance on purposive and snowball sampling, focus solely on Scrum framework, and the absence of quantitative data. The relatively small sample size of eight participants, while providing rich qualitative insights, limits empirical proof. This study could merely serve as a proof of concept for further empirical research on the interplay between LCNC, Agile and Artificial Intelligence.

6 References

Agile Alliance. (n.d.). *Principles behind the Agile Manifesto*. Agile Manifesto. https://agilemanifesto.org/principles.html

Alamin, M. A. A., Uddin, G., Malakar, S., Afroz, S., Haider, T., & Iqbal, A. (2023). Developer discussion topics on the adoption and barriers of low code software development platforms. *Empirical Software Engineering : An International Journal*, *28*(1), 4–4. https://doi.org/10.1007/s10664-022-10244-0

Almeida, F., & Carneiro, P. (2023). Perceived Importance of Metrics for Agile Scrum Environments. *Information (Basel)*, *14*(6), Article 327. https://doi.org/10.3390/info14060327

Aksenova, Z. A., Yashin, S. N., Markova, O. M., Chudaeva, A. A., & Alieva, P. R. (2024). Assessing the Impact of Digital Economy Programs on Alleviating Skill Shortages in the EU Labor Market for Digital Professionals. Journal of the Knowledge Economy. https://doi.org/10.1007/s13132-024-02202-6

Alt, R., Leimeister, J. M., Priemuth, T., Sachse, S., Urbach, N., & Wunderlich, N. (2020). Software-Defined Business: Implications for IT Management. *Business & Information Systems Engineering*, *62*(6), 609–621. https://doi.org/10.1007/s12599-020-00669-6

Al-Zewairi, M., Biltawi, M., Etaiwi, W., & Shaout, A. (2018). Agile Software Development Methodologies: Survey of Surveys. *Estonian Journal of Earth Sciences*, *67*(3), 74-. https://doi.org/10.4236/jcc.2017.55007

Ajimati, M. O., Carroll, N., & Maher, M. (2025). Adoption of low-code and no-code development: A systematic literature review and future research agenda. The Journal of Systems and Software, 222, 112300-. https://doi.org/10.1016/j.jss.2024.112300

Ang, R. J. (2021). Building Applications Using Low-Code and No-Code Platforms. Canadian Journal of Nursing Informatics, 16(3/4).

Appian Corporation. (2019, April 2). Independent study: 84% of firms with highest enterprise requirements use low-code development and see return-on-investment [Press release]. https://www.appian.com/news/news-item/independent-study-84-of-firms-with-highest-enterprise-requirements-use-low-code-development-and-see-return-on-investment/

Babaian, A. (2019). Becoming Agile with the Scrum Framework. Software Quality Professional, 22(1), 23–33.

Barkin, I., & Davenport, T. H. (2023). Harnessing grassroots automation. *MIT Sloan Management Review, 65*(1), 74-78. Retrieved from https://kuleuven.e-bronnen.be/scholarly-journals/harnessing-grassroots-automation/docview/2954922318/se-2

Bass, J. M. (2022). *Agile software engineering skills*. Springer Nature Switzerland AG. https://doi.org/10.1007/978-3-031-05469-3

Behutiye, W. N., Rodríguez, P., Oivo, M., & Tosun, A. (2017). Analyzing the concept of technical debt in the context of agile software development: A systematic literature review. Information and Software Technology, 82, 139–158. https://doi.org/10.1016/j.infsof.2016.10.004

Bhattacharyya, S. S., & Kumar, S. (2023). Study of deployment of “low code no code” applications toward improving digitization of supply chain management. *Journal of Science and Technology Policy Management*, *14*(2), 271–287. https://doi.org/10.1108/JSTPM-06-2021-0084

Bibik, I. (2018). *How to kill the Scrum monster: Quick start to Agile Scrum methodology and the Scrum Master role*. Apress. https://doi.org/10.1007/978-1-4842-3691-8

Bock, A. C., & Frank, U. (2021). Low-Code Platform. *Business & Information Systems Engineering*, *63*(6), 733–740. https://doi.org/10.1007/s12599-021-00726-8

Breyter, M. (2022). Agile product and project management: A step-by-step guide to building the right products right. Apress. https://doi.org/10.1007/978-1-4842-8200-7

Callinan, N., & Perry, M. (2024). Critical Success Factors for Citizen Development. *Open Journal of Applied Sciences*, *14*(4), 1121-1149.

Cassell, C. (2015). *Conducting research interviews for business and management students*. SAGE.
Chen, W.-E., Lin, Y.-B., Yen, T.-H., Peng, S.-R., & Lin, Y.-W. (2022). DeviceTalk: A No-Code Low-Code IoT Device Code Generation. *Sensors (Basel, Switzerland)*, *22*(13), 4942-. https://doi.org/10.3390/s22134942

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

Edison, H., Wang, X., & Conboy, K. (2022). Comparing Methods for Large-Scale Agile Software Development: A Systematic Literature Review. *IEEE Transactions on Software Engineering*, *48*(8), 2709–2731. https://doi.org/10.1109/TSE.2021.3069039

Elshan, E., Binzer, B., & Winkler, T. J. (2025). From Software Users to Software Creators: An Exploration of the Core Characteristics of the Citizen Developer Role and the Related Re- and Upskilling Programs: From Software Users to Software Creators. *Business & Information Systems Engineering*, *67*(1), 31–53. https://doi.org/10.1007/s12599-024-00915-1

Fikri, H. (2025). Interview transcripts: Low-code/no-code impact on agile teams [Unpublished raw data]

Fikri, H. (2025). Thematic coding analysis of LCNC interviews [Unpublished raw data].

Golov, R. S., & Myl'nik, A. V. (2023). Low-Code and No-Code Technologies in Designing Digital Infrastructure for High-Tech Enterprises. *Russian Engineering Research*, *43*(3), 334–335. https://doi.org/10.3103/S1068798X2304010X

Guthardt, T., Kosiol, J., & Hohlfeld, O. (2024). Low-code vs. the developer: An empirical study on the developer experience and efficiency of a no-code platform. *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems*, 856–865. https://doi.org/10.1145/3652620.3688332

Hagel, N., Hili, N., & Schwab, D. (2024). Turning Low-Code Development Platforms into True No-Code with LLMs. MODELS Companion '24 Proceedings, 876–885. https://doi.org/10.1145/3652620.3688334

Hanson, K. (2024). Beyond Lean. *Manufacturing Engineering*, *173*(5), 32–39.

Havelund, K., Steffen, B., & Margaria, T. (2021). Programming: What is Next. In *Leveraging Applications of Formal Methods, Verification and Validation* (Vol. 13036, pp. 195–201). Springer International Publishing AG. https://doi.org/10.1007/978-3-030-89159-6_13

Hoogsteen, D., & Borgman, H. (2022). Empower the workforce, empower the company? Citizen development adoption.

How No-Code/Low-Code Solutions Help IT Organizations Evolve. (2023). *Cio,* https://kuleuven.e-bronnen.be/trade-journals/how-no-code-low-solutions-help-organizations/docview/2763221844/se-2

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

Jaglan, N., Upadhyay, D., Bhattacharya, A., Dutta, S., Piuri, V., & Dutta, P. (2023). Decoding Low-Code/No-Code Development Hype—Study of Rapid Application Development Worthiness and Overview of Various Platforms. In *Innovations in Data Analytics* (Vol. 1442, pp. 419–427). Springer. https://doi.org/10.1007/978-981-99-0550-8_33

Jain, P., Sharma, A., & Ahuja, L. (2018). Software Maintainability Estimation in Agile Software Development. International Journal of Open Source Software & Processes, 9(4), 65–78. https://doi.org/10.4018/IJOSSP.2018100104

Kandaurova, M., Skog, D. A., & Bosch-Sijtsema, P. M. (2024). The Promise and Perils of Low-Code AI Platforms. *MIS Quarterly Executive*, *23*(3), 275-. https://doi.org/10.17705/2msqe.00098

Kanellopoulos, Y., & Yu, Y. (2015). Guest editorial: Special section: Software quality and maintainability. *Software Quality Journal*, *23*(1), 77–78. https://doi.org/10.1007/s11219-015-9270-x

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*, 196–205. https://doi.org/10.1109/CBI54897.2022.00028

Kass, S., Strahringer, S., & Westner, M. (2023). Practitioners' Perceptions on the Adoption of Low Code Development Platforms. *IEEE Access*, *11*, 1–1. https://doi.org/10.1109/ACCESS.2023.3258539

Khalajzadeh, H., & Grundy, J. (2025). Accessibility of low-code approaches: A systematic literature review. *Information and Software Technology*, *177*, 107570-. https://doi.org/10.1016/j.infsof.2024.107570

Kelly, W., Schröder, J., & Roock, S. (2019). Agile software architecture: Aligning agile processes and software architectures. Apress. https://doi.org/10.1007/978-1-4842-5168-3

Kirchhof, J. C., Jansen, N., Rumpe, B., & Wortmann, A. (2023). Navigating the Low-Code Landscape: A Comparison of Development Platforms. *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, 854–862. https://doi.org/10.1109/MODELS-C59198.2023.00135

Kok, C. L., Tan, H. R., Ho, C. K., Lee, C., Teo, T. H., & Tang, H. (2024). A Comparative Study of AI and Low-Code Platforms for SMEs: Insights into Microsoft Power Platform, Google AutoML and Amazon SageMaker. *Proceedings (IEEE International Symposium on Embedded Multicore/Manycore SoCs. Online)*, 50–53. https://doi.org/10.1109/MCSoC64144.2024.00018

Lebens, M., & Finnegan, R. (2021). Using a Low Code Development Environment to Teach the Agile Methodology. *Lecture Notes in Business Information Processing*, *419*, 191–199. https://doi.org/10.1007/978-3-030-78098-2_12

Lethbridge, T. C., Margaria, T., & Steffen, B. (2021). Low-Code Is Often High-Code, So We Must Design Low-Code Platforms to Enable Proper Software Engineering. In *Leveraging Applications of Formal Methods, Verification and Validation* (Vol. 13036, pp. 202–212). Springer International Publishing AG. https://doi.org/10.1007/978-3-030-89159-6_14

Liu, D., Jiang, H., Guo, S., Chen, Y., & Qiao, L. (2024). What's Wrong With Low-Code Development Platforms? An Empirical Study of Low-Code Development Platform Bugs. *IEEE Transactions on Reliability*, *73*(1), 695–709. https://doi.org/10.1109/TR.2023.3295009

Low Code/No Code Application Development - Opportunity and Challenges for Enterprises. (n.d.). https://doi.org/10.17762/ijritcc.v10i11.11038

López, L., Burgués, X., Martínez-Fernández, S., Vollmer, A. M., Behutiye, W., Karhapää, P., Franch, X., Rodríguez, P., & Oivo, M. (2022). Quality measurement in agile and rapid software development: A systematic mapping. The Journal of Systems and Software, 186, Article 111187. https://doi.org/10.1016/j.jss.2021.111187

Luo, Y., Liang, P., Wang, C., Shahin, M., & Zhan, J. (2021). Characteristics and Challenges of Low-Code Development: The Practitioners' Perspective. *ESEM 2021 - Proceedings of the 15th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 1–11. https://doi.org/10.1145/3475716.3475782

Lutwama, P., Dlulane, M., Pillay, T., Hassan, F. S., & Grobbelaar, S. (2024). AGILE: Advantages, Disadvantages, Enablers, and Barriers. *South African Journal of Industrial Engineering*, *35*(4), 66–76. https://doi.org/10.7166/35-4-3058

Maassen, M. A. (2018). Product development models in the IT sector-From Waterfall to Agile Project Management Model s in the case of AVIRA SOFT S.R.L. *Proceedings of the ... International Conference on Business Excellence*, *12*(1), 568–578. https://doi.org/10.2478/picbe-2018-0051

Martins, J., Branco, F., & Mamede, H. (2023). Combining low-code development with ChatGPT to novel no-code approaches: A focus-group study. Intelligent Systems with Applications, 20, 200289-. https://doi.org/10.1016/j.iswa.2023.200289

Matook, S., Wang, Y. M., & Axelsen, M. (2025). Experiential Learning for Citizen Developers. *Business & Information Systems Engineering*, *67*(1), 7–30. https://doi.org/10.1007/s12599-024-00921-3

Matvitskyy, O., Davis, K., & one more. (2024, October 16). *Magic Quadrant for Enterprise Low-Code Application Platforms* (ID G00804341). Gartner, Inc. https://www.gartner.com/doc/reprints?id=1-2J40TC35&ct=241017&st=sb

Maximini, D. (2018). The Scrum Culture: Introducing Agile Methods in Organizations (2nd ed. 2018.). Springer International Publishing AG. https://doi.org/10.1007/978-3-319-73842-0

Noll, J., Razzak, M. A., Bass, J. M., Beecham, S., Winkler, D., Méndez Fernández, D., Sarro, F., Turhan, B., Kalinowski, M., & Felderer, M. (2017). A Study of the Scrum Master's Role. In Product-Focused Software Process Improvement (Vol. 10611, pp. 307–323). Springer International Publishing AG. https://doi.org/10.1007/978-3-319-69926-4_22

No-Code, Low-Code Software Comes to HR. (2021). *HRNews,* https://kuleuven.e-bronnen.be/trade-journals/no-code-low-software-comes-hr/docview/2516867512/se-2

Papatheocharous, E., & Andreou, A. S. (2014). Empirical evidence and state of practice of software agile teams. *Journal of Software : Evolution and Process*, *26*(9), 855–866. https://doi.org/10.1002/smr.1664

Pavlic, L., Hlis, T., Hericko, M., & Beranic, T. (2022). The Gap between the Admitted and the Measured Technical Debt: An Empirical Study. *Applied Sciences*, *12*(15), 7482-. https://doi.org/10.3390/app12157482

Phalake, V., Joshi, S., Rade, K., & Phalke, V. (2022). Modernized Application Development Using Optimized Low Code Platform. *2022 2nd Asian Conference on Innovation in Technology (ASIANCON)*, 1–4. https://doi.org/10.1109/ASIANCON55314.2022.9908726

Picek, R. (2023). Low-code/No-code Platforms and Modern ERP Systems. 2023 International Conference on Information Management (ICIM), 44–49. https://doi.org/10.1109/ICIM58774.2023.00014

Putta, A., Paasivaara, M., & Lassenius, C. (2018). Adopting scaled agile framework (SAFe): a multivocal literature review. Proceedings of the 19th International Conference on Agile Software Development: Companion, 147763, 1–4. https://doi.org/10.1145/3234152.3234164

Prommegger, B., Arshad, D., & Krcmar, H. (2021). Understanding Boundaryless IT Professionals: An Investigation of Personal Characteristics, Career Mobility, and Career Success. *Proceedings of the 2021 Computers and People Research Conference*, 51–59. https://doi.org/10.1145/3458026.3462162

Razak, S. F. A., Ernn, Y. P., Yussoff, F. I., Bukar, U. A., & Yogarayan, S. (2024). Enhancing Business Efficiency through Low-Code/No-Code Technology Adoption: Insights from an Extended UTAUT Model. *Journal of Human, Earth, and Future*, *5*(1), 85–99. https://doi.org/10.28991/HEF-2024-05-01-07

Rethinking enterprise architects' roles for agile transformation. (2024). *Cio,* Retrieved from https://kuleuven.e-bronnen.be/trade-journals/rethinking-enterprise-architects-roles-agile/docview/3106294254/se-2

Ramesh, K. R., Divya, P. (2024) Revolutionizing Software Development: the Rise of No Code/low Code Development Solutions in Digital Era. International Journal For Multidisciplinary Research. https://doi.org/10.36948/ijfmr.2024.v06i02.16706

Rokis, K., & Kirikova, M. (2023). Exploring Low-Code Development: A Comprehensive Literature Review. Complex Systems Informatics and Modeling Quarterly, 2023(36), 68–86. https://doi.org/10.7250/csimq.2023-36.04

Rokis, K., Kirikova, M., Seigerroth, U., Sandkuhl, K., & Nazaruka, Ē. (2022). Challenges of Low-Code/No-Code Software Development: A Literature Review. In Lecture Notes in Business Information Processing (Vol. 462, pp. 3–17). Springer International Publishing AG. https://doi.org/10.1007/978-3-031-16947-2_1

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

Sanchis, R., García-Perales, Ó., Fraile, F., & Poler, R. (2020). Low-code as enabler of digital transformation in manufacturing industry. *Applied Sciences*, *10*(1), 12-. https://doi.org/10.3390/app10010012

Schmidt, C. (2016). Agile software development teams: The impact of agile development on team performance. Springer. https://doi.org/10.1007/978-3-319-26057-0

Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide: The definitive guide to Scrum: The rules of the game*. Retrieved from https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf

Shastri, Y., Hoda, R., & Amor, R. (2021). The role of the project manager in agile software development projects. Journal of Systems and Software, 173, 110871. https://doi.org/10.1016/j.jss.2020.110871

Stettina, C. J., van Els, V., Croonenberg, J., & Visser, J. (2021, June). The impact of agile transformations on organizational performance: a survey of teams, programs and portfolios. In *International Conference on Agile Software Development* (pp. 86-102). Cham: Springer International Publishing.

Tal, L. (2015). Agile software development with HP Agile Manager. Apress. https://doi.org/10.1007/978-1-4842-1035-2

Tang, L. (2022). ERP Low-Code Cloud Development. 2022 IEEE 13th International Conference on Software Engineering and Service Science (ICSESS), 319–323. https://doi.org/10.1109/ICSESS54813.2022.9930146

Thesing, T., Feldmann, C., & Burchardt, M. (2021). Agile versus Waterfall Project Management: Decision Model for Selecting the Appropriate Approach to a Project. *Procedia Computer Science*, *181*, 746–756. https://doi.org/10.1016/j.procs.2021.01.227

Uludağ, Ö., Putta, A., Paasivaara, M., & Matthes, F. (2021, June). Evolution of the agile scaling frameworks. In *International conference on agile software development* (pp. 123-139). Cham: Springer International Publishing.

Vincent P, Natis Y, Iijima K, Wong J, Ray S, Jain A, Leow A (2020) *Magic quadrant for enterprise low-code application platforms*. Gartner Report September 2020, Gartner

Waqas, M., Ali, Z., Sánchez-Gordón, M., Kristiansen, M., Mejía, J., Hernández Pérez, Y., Avila-George, H., Rocha, A., & Muñoz, M. (2024). Using LowCode and NoCode Tools in DevOps: A Multivocal Literature Review. In *New Perspectives in Software Engineering* (Vol. 1135, pp. 71–87). Springer. https://doi.org/10.1007/978-3-031-50590-4_5

Wolf, T., Schröder, J., & Roock, S. (2019). *Agile software architecture: Aligning agile processes and software architectures.* Apress. https://doi.org/10.1007/978-1-4842-5169-0

Wolff, I. (2019). Making In-House Apps with Low-Code, No-Code Platforms. *Manufacturing Engineering*, *163*(4), 58–67.

Woo, M. (2020). The Rise of No/Low Code Software Development—No Experience Needed? Engineering (Beijing, China), 6(9), 960–961. https://doi.org/10.1016/j.eng.2020.07.007

Zielinski, D. (2021). *No-code, low-code software comes to HR*. *HRNews*. Retrieved from https://kuleuven.e-bronnen.be/trade-journals/no-code-low-software-comes-hr/docview/2516867512/se-2

List of figures

List of tables

Appendices

Appendix 1: Link to all research files

https://drive.google.com/drive/folders/1fxKOWZYh472B_adBUxHcKyFizWl_ZDJa?usp=drive_link
Note. All files have been reviewed carefully to ensure anonymity of some research participants.