Process Optimisation Using a Low-Code Platform: Digitising a Concession Management Process

L. Smith[1] & M. de Vries[1*]

ARTICLE INFO

Article details
Submitted by authors 18 Jan 2025
Accepted for publication 7 Aug 2025
Available online 12 Dec 2025

Contact details
* Corresponding author
marne.devries@up.ac.za

Author affiliations
1 Department of Industrial and Systems Engineering, University of Pretoria, Pretoria, South Africa

ORCID® identifiers
L. Smith
https://orcid.org/0009-0002-9574-3204

M. de Vries
https://orcid.org/0000-0002-1715-0430

DOI
http://dx.doi.org//10.7166/36-4-3179

ABSTRACT

Low-code development platforms (LCDPs) recently emerged as a solution to the shortage of highly skilled professional software developers. Several inhibitors exist for adopting LCDPs at an enterprise, one of which indicates that LCDP adoption is inhibited because of a lack of use cases for LCDPs. A limited number of real-world implementations of workflow automations exist in which performance improvement has been demonstrated. The main contribution of this article is to present another use case of performance improvement, digitising mundane tasks, using Power Automate as an LCDP, in developing a concession management system at a home-schooling company.

OPSOMMING

Lae kode ontwikkeling platvorms (LKDPs) is ’n opkomende oplossing vir die tekort aan hoogs geskoolde professionele sagteware ontwikkelaars. Verskeie uitdagings belemmer die aanvaarding van LKDPs as oplossings in ondernemings, weens ’n gebrek aan gevallestudies. Verder is daar min werklike implementerings van werksvloei outomatisasies waar prestasie verbeterings gedemonstreer word. As hoof bydrae van hierdie artikel, demonstreer ons prestasie verbetering, wanneer alledaagse roetine take gedigitaliseer word met behulp van Power Automate as LKDP, in die ontwikkeling van ’n konsessie bestuur stelsel by ’n tuis-onderrig maatskappy.

1. INTRODUCTION

The landscape of digital technologies is constantly changing [1]. Low-code development platforms (LCDPs) are tools that have garnered interest in both academic and industry environments [2, 3], as they address the shortage of highly skilled professional software developers, offering end users with limited programming background the opportunity to contribute to software development processes that are focused on the business logic of applications [4]. A growing number of vendors responded to the need to boost productivity, promising limited coding during software development [5]. Relating to the LCDP and agile development context, this section presents our main research question and the benefits of pursuing this study.

1.1. Low-code development platforms and agile software development context

Low-code development platforms (LCDPs) are software tools that allow users, also called citizen developers [6], to create business programs using a graphical interface with minimal programming skills [7]. These platforms intend to *increase productivity*, reduce costs, and improve the ability to adapt software to rapidly changing requirements [8]. Companies increasingly consider low-code technology owing to the potential productivity increase when developing information systems [9]. With these platforms, developers can easily build custom business programs by using pre-built components [10]. These platforms are accessible as free or low-cost offerings [11], providing a subset of frequently used functions in the form of drop-down menus in the graphical user interface, where essential features are saved as modular sections for reuse [12]. The flexibility of low-code helps to keep the software sustainable [13], but also to optimise processes through automation, improving communication via the integration of applications [12].

Agile is an iterative approach to software development that has grown dramatically [6], emphasising the value of competent people when they collaborate as self-organising teams, incrementally developing software, which encourages rapid and flexible responses to change [7, 9]. Some agile practices contribute not only to faster solution development, but also to more sustainable software engineering [6, 10, 11] – for example, accepting that requirements might change, engaging in rich communication and collaboration, emphasising simple design, using minimal documentation, preventing early defects, testing regularly, and managing the product life cycle sustainably [6].

In the tertiary education sector, lecturers in information system development started to experiment with the combined use of an LCDP (Microsoft Power Apps) with an agile methodology [12]. Feedback from students who used Power Apps as part of an information systems design course indicated positive perceptions, but also identified some drawbacks, such as the fact that coding skills are still highlighted as one of the conditions for using Power Apps [13]. Low-code platforms are easy to learn, they are model driven, and they take advantage of cloud infrastructures, automatic code generation, and graphical abstractions to develop functioning applications [14]. Yet, according to Reijers and Van der Aalst [15], there are only *limited case studies* on real-world implementations of workflow automations, demonstrating *performance improvement*. Also, Käss *et al.* [16] indicate that there are several inhibitors of adopting LCDP in an enterprise, one of which is that LCDP adoption is inhibited by a *lack of use cases for LCDPs*. Based on the gap in the research, the next section presents our main research question and study objectives.

1.2. Research question, objectives, contribution, and benefits

> Given the lack of existing real-world case studies, the primary research question is: *How could a low-code development platform be used, combined with software development practices, to digitise mundane tasks and improve performance at a home-schooling company?*

The main objectives of this article also provide the focus of our contribution: (1) to provide a practical demonstration of using a low-code development platform, (2) combined with some software development techniques, and (3) demonstrating improved performance at a real-world enterprise. Reflecting on the success of the project, we provide practical insights gained from it. We followed a design-based approach, as discussed in Section 3, to answer our research question. Our design-based approach provided the context to share practical insights into using a low-code platform in a real-world context, rather than following a quantitative hypothesis-based approach.

The remainder of the paper is structured as follows. Section 2 provides the background to the project and the problem experienced at a real-world company, elaborating on the theoretical context that applies in solving the problem. Based on the research gap identified in section 2.4, section 3 defines our methodology

in developing a digitised solution. Section 4 applies the methodology presented in Section 3, while Section 5 presents the testing and evaluation results, followed by a discussion in Section 6 of the practical insights, difficulties, and implications for researchers and practitioners. We conclude with the research's limitations and recommendations for future research in Section 7.

2. BACKGROUND

We have provided background to the project and product context in section 2.1, describing a real-world company, EduC (a fictitious name to preserve the company's confidentiality). We also present the main problems experienced at EduC in section 2.2, while section 2.3 provides the theoretical context of workflow systems and the emergence of LCDPs. Section 2.4 presents the literature on scholarly works that include low-code development tools combined with software development practices, and highlights a research gap.

2.1. Project and problem background

In 2019, EduC Campus entered the education industry with a mission to provide additional support to Grades 4 to 12 learners striving to achieve their National Senior Certificate. Recognising the need for accessible and affordable tutoring services, EduC began to offer online distance learning to learners. EduC's commitment to academic excellence is reflected in the educational pathways that they offer to over 32,000 learners in four departments: Home, Class, College, and Business. EduC's project was begun with only the Home Department to reduce the scope of the project. The Home Department provides services to support Grades 10 to 12 learners in their studies from the comfort of their own homes; these services are aligned with the South African curriculum assessment policy statements and the standards of the Independent Examinations Board.

One area of concern for EduC was the management of four types of concession: immigrant status, appeal, ad hoc, and accommodation. *Immigrant status* concessions allow for an exemption from having a first additional language subject and for a different subject to be substituted, with Business Studies being a possible alternative. An *immigrant status* concession is aimed at learners who are not proficient in Afrikaans. *Appeal* concessions provide a recourse for clients who are dissatisfied with the outcome of an application. Ad hoc concessions are granted in emergencies, such as when a client is temporarily unable to write because of an arm injury. *Accommodation* concessions are aimed at removing barriers to learning and providing special privileges for examination purposes.

Evaluation of a concession application, required extensive communication between the client and the EduC employee, which was a largely manual process. Storing data in Excel and relying on simultaneous data entry by two employees increased the likelihood of human error in the data. In streamlining the concession evaluation process, EduC needed a feasible solution that was cost-effective and could be implemented using Power Apps, for which the company has a licence. The solution had to be tailored to EduC's unique requirements, using Power Apps and SharePoint. The development had to proceed in the context of the concession department, which consisted of three team members, one of whom was a developer with extensive knowledge of Power Apps and who provided active mentorship to the main author of this paper. The developer also provided the main author with access rights to SharePoint, which the developer had already configured.

In respect of scale, EduC receives an estimated 300 concession applications per year. Each concession instance follows a process with multiple stages, each of which requires an average time for completion. By implementing a more efficient concession management system, EduC needed to reduce the total duration of the evaluation process, thereby enabling the company to focus more closely on providing quality education and support to its learners.

In summary, *the main problem* experienced by EduC was that the current process was time-consuming because it relied on manual procedures. A more sophisticated system was required to streamline the concession management process, reducing the margin of error and enhancing overall efficiency. A more sophisticated system would enable the organisation to meet the needs and expectations of its stakeholders and minimise the workload of its employees.

2.2. Improvement objectives

Process performance – that is, the end-to-end process time – is a key performance indicator for EduC. To determine the time needed for two employees to process the year's *accommodation* concessions, an average estimate was used, as indicated in Table 1, using historical data. Applications for students in Grades 4 to 9 are assessed internally by EduC's committee and a psychologist. For Grades 10 to 12, an external entity, the South African Comprehensive Assessment Institute (SACAI), performs the evaluation part of the process. EduC processes feedback from both the committee and SACAI to provide applicants with comprehensive feedback. The project comprises two processes: one for Grades 4 to 9, and a second for Grades 10 to 12. This study concerned the improvement of only the *accommodation* concession type, which itself has demanded about three months of work by two staff members. In addition to managing *accommodation* concessions, employees have also had to manage other concession types, increasing their workload. Since the project had to compare the duration of manual-driven tasks before automation and after automation, we refer to six automation phases in the article:

- Phase 0: Evaluate application completeness
- Phase 1: Capture data
- Phase 2: Prepare documents for committee or SACAI evaluation
- Phase 3: Process evaluation results received from the committee or SACAI
- Phase 4: Update the system with the committee or SACAI results
- Phase 5: Send feedback to the student

The time constraint has been a significant factor for the two employees, as most concession applications are submitted early in the year, before the first tests and assignments. As shown in Table 1, using an eight-hour working day, this has placed considerable pressure on the employees to complete the workload prior to learner assessment. Furthermore, the number of students requiring concessions increases each year, exacerbating the issue. To ensure a sustainable solution, EduC required a more efficient and scalable solution.

Table 1: Time required to accomplish a task during the internal process

Phase	Task and quantities
Phase 0	Work through the application to see whether the application is complete
Phase 1	Enter the data into the system Number of applications: 300 complete and 100 incomplete
Phase 2	Prepare documents to be evaluated by EduC's Committee or SACAI Number of applications: 200
Phase 3	For Grades 4-9
	Prepare the document after it has been evaluated by the committee Number of applications: 50
	For Grades 10-12
	The time it takes to prepare the document received from SACAI Number of applications: 150
Phase 4	Update system after SACAI's and committee's evaluation process Number of applications: 200
Phase 5	Send feedback to clients

Grades 4-9

Process	Minutes	Applications	Total minutes
Phase 0	12	106	1,272
Phase 1	20	66	1,320
Phase 2	60	66	3,960
Phase 3	60	66	3,960
Phase 4	40	66	2,640
Phase 5	10	66	660
			13,812

Total hours	Total days
230	29

Grades 10-12

Process	Minutes	Applications	Total minutes
Phase 0	12	274	3,288
Phase 1	20	211	4,220
Phase 2	60	211	12,660
Phase 3	60	211	12,660
Phase 4	40	211	8,440
Phase 5	10	211	2,110
			43,378

Total hours	Total days
723	90

Synthesis

Total working days:	119
Total working weeks:	23.8
Total working months:	6.0
Working months for two employees:	3.0

2.3. Theoretical context

The project presented in this article opened up the opportunity to experiment with an LCDP, since Power Apps was mandated by EduC. In addition, we could experiment with the *workflow* and *process automation* capabilities that Power Automate offered. In this section, we present the background to workflow systems and LCDPs.

2.3.1. Flow charts, workflows, and product development workflow

Flow charts provide an overview of functional activities that are performed at an enterprise in a particular sequence, whereas *workflows* expand on the business logic of a flow chart that supports a workflow programmer in automating a workflow [17]. A *product development workflow* involves robotic process automation, integrating different applications, selecting appropriate user interface elements, and manipulating data [18].

2.3.2. Low-code development platforms for robotic process optimisation

Sahay *et al.*'s [4] taxonomy of LCDP features emphasises the comprehensiveness of these platforms, offering different *categories of features*: graphical user interfacing, interoperability with external services, security support, collaborative development support, reusability support, scalability, business logic specification mechanisms, application build mechanisms, deployment support, and different kinds of supported application. Gürcan and Taentzer [19] indicate that Sahay *et al.*'s [4] taxonomy failed to highlight *data modelling* features, whereas *interoperability with external services* failed to distinguish between pre-built external services and custom services, given the major differences in the development of these connections.

Maximising their low-code capabilities, LCDPs are often used in combination with multiple technologies offered by different vendors [14, 20, 21]. Vincent *et al.* [21] use a *different taxonomy* from Sahay *et al.*'s [4] to differentiate between *low-code platforms* that are offered by vendors. For instance, low-code application platforms focus on software application development, with Mendix as an example, whereas citizen automation and development platforms (CADP) focus on citizen automation, such as Microsoft Power Automate with Power Apps [21]. Integration platform as a service technologies run in the cloud, without the need of managing local servers such as Power Automate [22], whereas robotic process automation (RPA) technologies focus on process automation such as UiPath and Blue Prism [21].

2.3.3. Power Automate

Power Automate, formerly known as Microsoft Flow, is primarily a cloud-based workflow engine that can be used to automate common business processes or sequences [22]. A Power Automate solution includes multiple *flows*, defined as logical groupings of connectors, triggers, conditions, and actions [22]. *Branching* is a concept that is used to indicate that different sets of action may happen in which branches are activated via different conditions *or* executed in parallel [22]. A *gateway* is required when an application service such as Power Automate must access data sources located in an on-premises environment such as a SharePoint server.

2.4. Related work

The number of *low-code platforms* (LCPs) are still emerging, and navigating the low-code landscape is a challenge [23]. As indicated before, Vincent *et al.*'s [21] taxonomy is useful to differentiate between LCPs that are offered by vendors, since low-code application platforms focus on software application development, with Mendix as an example, whereas CADPs focus on citizen automation, such as Microsoft Power Automate [21]. For this study, Microsoft Power Automate with Power Apps was mandated as the CADP to develop a solution. Furthermore, the main researcher applied several software development practices, including agile practices, as discussed in section 3, to support the iterative requirements elicitation, building, and testing of the solution.

To contextualise and compare our approach with the body of knowledge, we had to search for other studies that used CADP. Frank *et al.* [5] identified 10 relevant LCPs that are *marketed as LCPs* and are *accessible*, analysing and comparing these platforms to capture the common characteristics and the specific features of individual tools. Examples of LCP vendors that were excluded due to their *inaccessibility*, when an in-depth analysis of the platforms could not rely on marketing material provided by the vendor itself, were Salesforce, Google, Oracle, and ServiceNow [5].

Four prototypical categories were suggested as useful for providing an initial orientation in selecting 10 LCPs from an initial list of 30 potential candidates; other considerations were used that included some of the LCPs, such as having a *considerable market position* and being *established* for a number of years [5]. The four prototypical categories with their associated LCPs are as follows:

1 *Basic data management platforms.* LCPs in this category view and edit data that are organised in GUIs according to the user's needs [5]. LCPs: QuickBase, TrackVia.

2 *Workflow management systems.* These LCPs focus on the dynamic aspects of an application rather than on data management, provide a visual design of workflows, and include support for workflow execution and third-party connections. The availability of source code editors is generally not a necessary feature for using the platform [5]. LCPs: Bonita Studio, Creatio Studio.

3 *Extended, GUI- and data-centric integrated development environments (IDEs).* These LCPs resemble regular IDEs that require familiarity with programming languages, with extensive support to write source code or specific support to integrate source code files from external sources. Platforms in this category do not have a specific focus on either data or workflow management [5]. LCPs: Mendix Studio (Pro), WaveMaker Studio, Zoho Creator.

4 *Multi-use platforms for business application configuration, integration, and development.* These LCPs aggregate the preceding functionalities in the other three categories, with the focus on integrating and developing a variety of application artefacts – that is, integrating several internal and external development artefacts [5]. LCPs: Microsoft Power Apps, Appian, Pega Platform.

Some studies include *how-to* knowledge for using low-code development platforms, with step-by-step detail to create an application using Power Apps [24], and with examples of how the different MS Power Platform products could be used in combination [25]. They demonstrate small-scale single-process automations [26], provide brief explanations of how to create an application using the *Canvas app* development approach [27], and also spell out some logical development steps associated with user interface designs. However, the designs are presented in Finnish [28].

Although the body of knowledge presents some knowledge about using a low-code development platform in practice, more research related to the low-code trend in general is needed that indicates appropriate circumstances in which the use of these platforms could help enterprises to improve their productivity [8].

3. METHODS

We followed a *design-based* methodology – that is, a researcher learns through making and producing new technology products, also called *artefacts* [29]. One of the valid types of artefact includes an *instantiation*, a "working system that demonstrates that constructs, models, methods, ideas, genres or theories can be implemented in a computer-based system" [29]. For our study, the instantiation is a re-designed concession management system (CoMS) as the computer-based system, developed using a *low-code platform*. In accordance with the guidelines on *design science research* [30, 31], we applied five design cycle phases:

- *Identify problem and motivate.* The problem of *inefficient operations and management of concessions* is presented in section 2.1.

- *Define objectives of a solution.* The objectives of the solution are given in section 2.2, highlighting that the *end-to-end process time* was a critical metric (or key performance indicator) for improvement at EduC.
- *Design and development*. Section 3 elaborates on the participants who were involved and their levels of experience.
- *Demonstration.* During the demonstration, researchers need to show the usefulness of the artefact in solving one or more instances of the problem [30]. Section 4.2, together with Appendix A, provides details of the construction of the CoMS, showing the flow of work implemented by the low-code application.
- *Evaluation.* The evaluation phase has to "measure how well the artefact supports a solution to the problem" [30]. Section 5 elaborates on using unit tests, functional tests, system quality tests, and acceptance tests, indicating that EduC's main concern with the current time-consuming process was solved.

The main participants in the development of the CoMS were the following:

- The product owner, also acting as the primary end user, was the main data-gathering source during requirements elicitation.
- The citizen developer (CD), also the main author of this study, had some undergraduate introductory training on information system design and the low-code platform Mendix.
- A senior developer (SD) provided expertise to resolve one of the errors. The SD had nearly 17 years of experience working as software engineer in multiple South African industries, including government, financial services, food production, and education technology. Over the past three years the SD has digitised several processes in EduC, using Microsoft Power Automate.
- A junior developer assisted in providing internet search terminology when the CD experienced technical problems or errors. The junior developer's professional journey started around 2006, focusing primarily on reconciling high volumes of transactions from various contributors each month and gaining experience in error-handling. In 2000 she completed a Microsoft Office course that focused on using Microsoft Excel, and in 2014 she extended her Excel skills, completing Excel Visual Basic for Applications (VBA). She used Google, YouTube, and ChatGPT to train herself to use the Microsoft Power Suite.
- End users, two qualified industrial engineers, were involved in user acceptance testing.

Since EduC had already embraced an agile way of working during software development, this study also provided transparency in applying selected software development practices, designing a solution to address the problem of *the inefficient operations and management of concessions*. Using an incremental approach to a complete product, the first version of the product is known as the minimum viable product (MVP) [32]. Whereas a prototype is a pre-production representation of some aspect of a final design [33], the MVP has to be *an increment of the product* that is implemented in practice [32].

4. SOLUTION REQUIREMENTS AND SOLUTION CONSTRUCTION

The preliminary design stage focused on analysing the existing process, indicated in **Figure 1**, which uses the Business Process Model and Notation (BPMN) version 2.0.2 standard [34, 35, 36], and which allows the user to explore various concepts and ideas for a proposed solution.

4.1. Solution requirements

Analysing the process shown in **Figure 1**, we highlighted tasks that did not require human decision-making, using a dotted-red border and indicating opportunities for partial automation. In general, tasks earmarked for automation included sending customised emails, checking whether mandatory fields were filled in, capturing data, retrieving data, updating data, and formatting documents. Automating these tasks would free up time for employees to focus on tasks that required higher cognitive thinking. The message flows to and from SACAI, without a dotted-red border, indicated that communications between the concession coordinator and SACAI would not be automated with the solution construct; that is, the concession coordinator would still generate a PDF form that was sent manually via e-mail to SACAI, and SACAI would still send their decisions via e-mail.

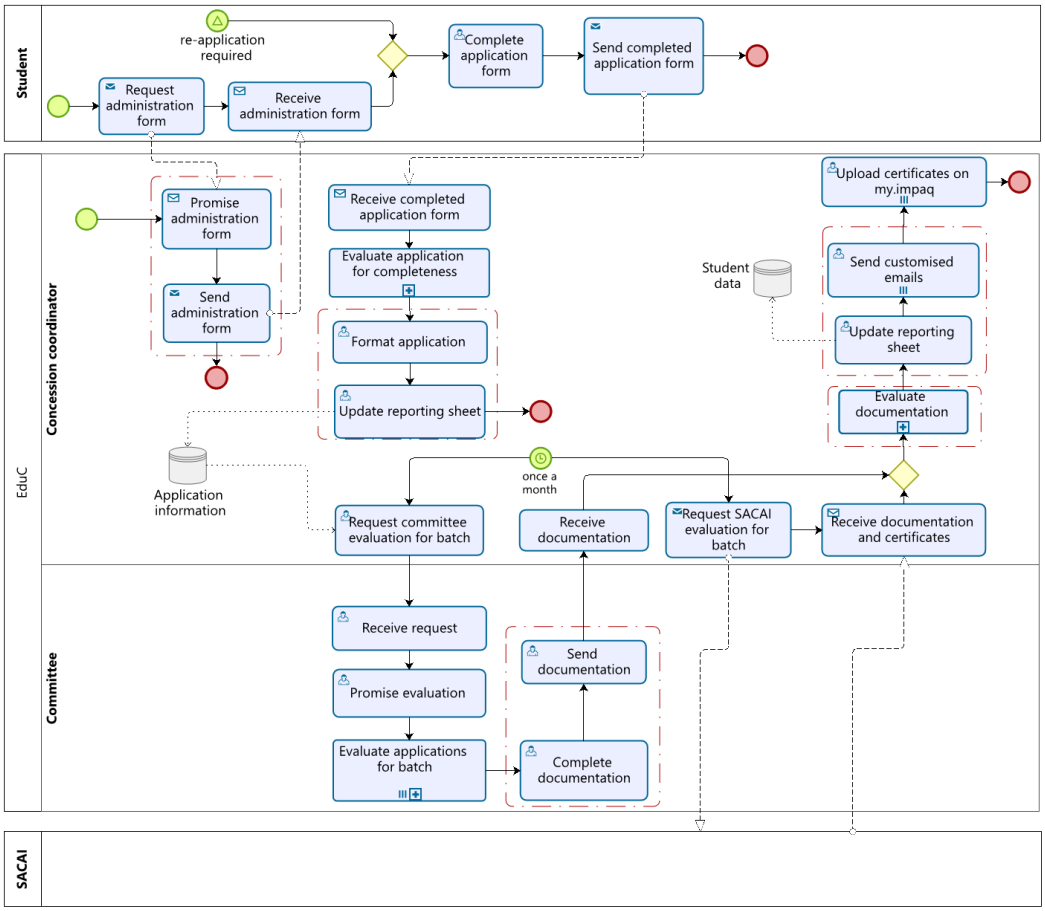

Figure 1: Repetitive tasks identified for automation, highlighted in red

Acknowledging the iterative process of requirements elicitation [37], a list of key requirements were extracted. One of the mature requirements elicitation techniques, the *interview* – still often used during requirements elicitation [38] – was also applied in our study, especially during the problem investigation phase of the project. Another frequently used requirements elicitation technique is *questionnaires* [39], which we used to extract the automation requirements, both functional and non-functional. Ameller *et al.* [40] emphasise the need to elicit both functional and non-functional requirements during the software requirements elicitation process. We used a PIECES (performance, information, economic, control, efficiency, and services) checklist and Leffingwell's [37] list of non-functional requirements. The PIECES checklist is useful in analysing an information system and identifying areas that are deficient [41, 42], whereas Leffingwell [37] presents a list of common non-functional requirements that serve as a checklist to extract system requirements from system users.

Pacheco *et al.* [38] indicate that *user stories* are often used in agile software development approaches. The user story format, specified by Leffingwell [37], helped us to focus on value-adding requirements – that is, "As a <role> I <need to perform an activity> so that <business value is obtained>". In addition, emerging requirements were classified according to their priority, using the *MoSCoW technique*. MoSCoW, an acronym for must-have, should-have, could-have, and want-to-have, is a technique that is used to prioritise and categorise software requirements, differentiating between essential and additional requirements [43, 44]. The technique is particularly useful when needs are not well-defined in the early stages of projects [45], as it allows for reprioritisation [46]. The must-have priority level encompasses crucial features, the absence of which would lead to a failure to address critical user expectations. The should-have category comprises desirable capabilities for a computer system. In contrast, the could-have category includes features that are worth considering but that are less important than the should-have group. Last, the want-to-have group comprises optional features that could be considered if additional finances were available [47].

Table 2 presents a *sample* of must-have requirements that EduC considered, applying the MoSCoW requirements prioritisation. The functional requirements were defined in user story format – that is, "As <role> I <need to perform an activity> so that <business value is obtained>". Two main roles were identified for the CoMS: the *student* and the concession *evaluator*.

Note that the requirements in Table 2 are context-specific. Adherence to the code "Free" in Table 2 depends on the type of Microsoft licensing package stipulated by the Microsoft Power Platform Licencing Guide [48] – that is, if the enterprise bought *select* Microsoft 365 licences, some of the Power Apps and Power Automate functionalities would be free of charge (for example, manually triggering cloud flows, not desktop flows), as long as standard connectors were used, such as SharePoint.

Table 2: Sample of must-have requirements for the CoMS

Code	User stories for functional requirements
File-upload	As *evaluator*, I need the system to allow file uploads after concession approval to ensure data integrity.
Auto-follow-up	As *evaluator*, I need the system to facilitate automatic interaction with students to address missing information.
Performance	The system should be more time-efficient than the current process for improvement.
Free	The system should not require additional funds because they can invest in another project.

4.2. The concession management system construction

The Microsoft Office Suite includes a number of productivity applications such as Word, Excel, OneDrive, SharePoint, Microsoft 365 Teams, Yammer, Azure Stream, Dynamics 365, Power Platforms (with Power BI, Power Apps, and Power Automate), and many others [49].

Microsoft Power Automate was used primarily to develop a software solution for EduC. The user-friendly interface enables citizen developers to create automations that can span multiple systems and applications with the objective of replacing repetitive actions with an automated flow [49].

The main constructs of the CoMS consist of multiple workflow scenarios. Power Automate allowed for a visual interface for low-code development to support each scenario. This section presents the main workflow scenarios that were built using Power Automate, combined with Microsoft Forms, Teams, and SharePoint. EduC's current platform provides the student (as applicant) with a link to the Microsoft *concession application form*. The applicant has to complete the form, which guides the applicant to attach the supporting documents that EduC requires. Appendix A provides more detail on the constructional parts of the CoMS.

5. TESTING AND EVALUATION RESULTS

A number of tests were performed on the CoMS:

- *Unit tests* were performed using the built-in *flow-checker* button before executing the actual program. Tests were also performed to ensure that the system-generated results were aligned with the expected results.
- *Functional tests* assessed functionality by testing story-acceptance, using the user stories, classified as functional requirements. (See the sample of functional requirements in Table 2.)
- *System quality tests* focused on testing the non-functional requirements. (See the sample of non-functional requirements in Table 2.) Additional performance tests were conducted; for example, various file types of different sizes were uploaded to the system to test for potential difficulties with file size and file types. No additional security tests were conducted, relying instead on Microsoft's access security features.
- Finally, *system acceptance tests* started with walk-through testing with the end user of the CoMS, in which the end user asked questions and provided feedback and critique. The end user also performed exploratory testing, scenario testing, usability testing, and user acceptance testing. Some functional deficiencies were identified and addressed by the CoMS developer.
- The walk-through testing demonstrated *six phases* of automation in managing concessions, which have already been introduced in section 2.2.

EduC's main concern with the current process was that it was time-consuming. Table 3 illustrates the improvement, showing that the time needed for the process was reduced by 92%. Note that we used calculated/inferred data rather testing data. Note too that Phase 0 (associated with the sub-process "Evaluate application for completeness" in **Figure 1**) is also a mundane task, and so it was not earmarked for automation in the scope of this study.

Table 3: Walk-through testing and evaluation

Grades 4-9

Process	Minutes	Applications	Total minutes
Phase 0	12	106	1,272
Phase 1	0	66	0
Phase 2	0	66	0
Phase 3	0	66	0
Phase 4	0	66	0
Phase 5	0	66	0
Total 1			*1,272*

Grades 10-12

Process	Minutes	Applications	Total minutes
Phase 0	12	274	3,288
Phase 1	0	211	0
Phase 2	0	211	0
Phase 3	0	211	0
Phase 4	0	211	0
Phase 5	0	211	0
Total 2			*3,288*

Project results	
4,560	Total minutes of new process (Total 1 + Total 2)
57,190	Total minutes of original process (see Table 1)
8%	Percentage time used by manual labour for new process (4,560/57,190)
92%	Time automated (for internal phases)

6. PRACTICAL INSIGHTS, PROBLEMS, AND IMPLICATIONS

The project provided an opportunity to experiment with Power Automate's functionalities, combined with Microsoft Forms, Teams, and SharePoint, in automating some of the tasks that form part of concessions management at EduC. We now present our main experiences in the form of practical insights, problems or difficulties, and implications, which we also compare with the body of knowledge.

6.1. Practical insights

We offer practical insights into our software development approach, self-training, and access to experienced developers for support.

1. Using pragmatic software development techniques. In alignment with the agile way of working at EduC, we had to use software development practices that were encouraged by the concession management team. Requirements elicitation, including identifying opportunities for automation, was facilitated primarily by drafting a flow chart, more specifically a collaboration diagram, using the modelling software Bizagi. We used a standard notation language, BPMN, to communicate with the stakeholders about their existing process. Several other research scholars [26, 28, 50] support the use of flow charts during the requirements elicitation phase of low-code development. A number of agile software development techniques were useful, including rich communication and collaboration between the developer and the end users, with minimal documentation and regular testing, using the built-in flow-checker of Power Automate. User stories assisted in tracking user requirements and prioritising requirements using the MoSCoW technique. User acceptance tests indicated that users were satisfied with the initial version of the CoMS as an MVP. Although minimal documentation was used to provide specifications for the CoMS, the visual representation of the flow-logic, exemplified in Figures 2 to 6 in Appendix A, already provided insight into the procedural logic of the CoMS, enabling its extension for future versions.

2. Self-training. Once we had identified tasks that could possibly be automated, the citizen developer explored the automation functions offered by Power Automate and identified a number of videos that provided useful guidance, including:

- A demonstration of how to save Microsoft Forms to Microsoft SharePoint [51]
- A Microsoft Teams approval workflow demonstration, using Power Automate [52, 53, 54]

Based on the experience of this project, the step-by-step online videos about Power Automate were sufficient guidance for automating processes such as sending emails, storing data, and evaluating

documents. The learning courses provided by Power Automate were not needed, confirming the claim that low-code development tools are easy to learn and require few or no programming skills. Although our experience indicated that guidance from an experienced developer is recommended for workflow creation, the online community forum [55] can also be consulted in resolving errors. In support of our experience, Vartianen [28], using Power Automate to develop a prototype, indicated that code snippets from diverse online repositories, including Google, YouTube, and ChatGPT, assisted his development efforts.

3. Experienced developer support. A key insight from the success of this project is that the citizen developer should have access to knowledgeable developers who have used Power Apps before. This study's researcher did not receive active demonstrations from the experienced developers. Rather, advice about appropriate search terms for certain functions or typical errors provided sufficient assistance to proceed with self-help, using a Google search. Only one of the errors could not be resolved; it required sending a visual image of the error to the senior developer.The body of knowledge corroborates our finding that the citizen developer would need some background to or experience of information system development; for example, when developing a Power Apps application, "experience is heavily influenced by prior knowledge and expertise in software development" [27], and when developing an application with Power Pages, "low-code development still requires knowledge about common architectures for building more advanced systems and databases" [56].

6.2. Problems and implications

The complexity and integration points of the application might pose additional difficulties. The CoMS was developed by using only Microsoft products – that is, Power Automate, Forms, Teams, and SharePoint. Another study that used Power Apps in combination with SharePoint reported difficulties when Power Apps had to search for data sets in SharePoint [57]. Palmer [57] also warns that Power Apps supports only a handful of outside data sources, which could make third-party integrations problematic.

Since the main developer was involved only in developing the MVP, some problems might still emerge in the future. As an example, Al-Qassab [24] reported his experience of Microsoft deleting some features from the previous released version of Power Apps. Evans and Petersson [56] indicate that the Power platform goes through massive changes twice a year that call for "updating every system connected to Power Platform to work with the new updates[,] which can be a massive maintainability project". Although maintainability is an issue with all low-code platforms, Ferdous [58] indicates a practical approach to platform updates, advising the low-code developer to search forums or discussion platforms when encountering issues. Yet, when no such guidance is available, developers must rely entirely on trial-and-error and on their own logical reasoning to resolve the issue [58].

Evans and Petersson [56] used the MS Power Platform to develop a website using Power Pages. They highlighted several difficulties, comparing low-code developing with traditional programming: (1) the limited ability to customise generated code, which hampers maintainability, flexibility, and testability; (2) difficulty in following the flow-logic, where hard-to-read code leads to "developers working with low-code tools with limited knowledge having to work with a large and hard-to-read codebase, which becomes counterproductive when it comes to using low-code tools"; and (3) reusability outside the platform is low, since "the functions created in the platform stay in the platform and the code generated is inaccessible" [56].

Power Automate now supports versioning for solution-aware cloud flows. As a team develops, version history is stored in Dataverse, allowing users to view past versions and to restore them if needed [59]. When it is ready, change can be committed to a branch in the connected Azure DevOps Git repository [60]. Teams can use part of Azure DevOps called Pipelines in Power Platform to use in a test or a production environment for a release [60].

7. LIMITATIONS, RECOMMENDATIONS, AND CONCLUSIONS

EduC required an automated solution owing to their need to reduce the manual workload in their current process. The manual aspect of the process was susceptible to human error, as staff had to record and save information manually using Microsoft Excel. The continual back-and-forth communication via email between the students and EduC's employees amplified the inefficiencies that were ingrained in the system. Using cloud flows of Power Automate, Microsoft Forms, Microsoft Teams, and SharePoint, the first version of the new CoMS was developed and implemented, resulting in a 92% reduction in time consumption and in

the manual labour in EduC's concession department. The study had two kinds of limitation: (1) method limitations, and (2) generalisability and scalability limitations.

7.1. Method limitations

We followed a design-based approach to develop a single artefact, the CoMS, to solve a problem. Since we had to develop a solution to a real-world problem, we had to use a pragmatic approach in the application of methods, using software development practices that were also encouraged by the concession management team – for example, using an agile approach and flow-charting to understand the existing process and to discuss problematic areas. Other researchers, using some Power platform products, used different software development techniques, such as using Azure DevOps tools and techniques [24] and value stream mapping [61]. We do not claim that, in our study, our selection of techniques, such as process flow charts, user stories, MoSCoW requirements prioritisation, and testing techniques, was the main contributing factor in the success of the project.

Change management was a facet that was partially incorporated by following a highly participative and incremental process in developing and testing the CoMS and delivering comprehensive training documentation on using the CoMS. After user acceptance, EduC did not immediately deploy the CoMS because of capacity constraints. However, EduC identified a suitable individual to drive the implementation and system monitoring from 2026. Although our study reports on inferred improvements, a real-world implementation would strengthen our claims.

Similar to the study of [62], future studies need to perform empirical comparisons of different software development approaches when combined with low-code development projects. Another limitation is that the main researcher actively participated in the development of the artefact, partially addressing the gap in the literature: the lack of use cases for LCDPs. The reported practical insights, difficulties, and implications are problem-context-sensitive, and focus primarily on the experiences of the main researcher.

7.2. Generalisability and scalability limitations

One of the main limitations of our study is the specific Microsoft technology stack that we used of Power Automate, Forms, Teams, and SharePoint, which limited the generalisability of the study. A recent study by Vartianen [28] indicated a different Microsoft technology stack (Power Automate, Forms, Power Automate Desktop, and Outlook), in which the low-code application could *not* be implemented owing to licensing constraints – that is, "Despite possessing licences allegedly covering all Microsoft tools, it became known that Power Automate Desktop necessitated separate purchase". Regarding *scalability*, the CoMS was implemented to digitise the tasks associated with 300 concession applications per year. The study excluded scalability tests, which might become problematic as EduC extends its customer base. Although LCDPs may be an interim solution in addressing efficiency issues, the solution may only be scalable on the basis of the licensing package that was bought.

7.3. Conclusions

Concluding with our main research question, our design-based study provided another demonstration of a successful low-code development initiative, combined with software development practices, to digitise mundane tasks, improving process performance at a real-world home-schooling company.

Since LCDPs are still developing, additional use cases would be required to create additional confidence in adopting these platforms as reliable and maintainable solutions for real-world companies. Although EduC decided to use the Power Platform, based on existing Microsoft licensing decisions, various other conditions need to be considered when an enterprise considers adopting an LCDP. Käss *et al.* [63] have already derived archetypes for LCD adoption or non-adoption that might be useful to practitioners when they select suitable cases for experimentation, based on existing LCDP decision drivers in their enterprise context. We suggest that *maintainability* should be used as an important entry requirement for selecting low-code platforms. Although use cases contribute to the practical use of these platforms in speeding up development, Bock and Frank [8] suggest some inspiring research topics related to the low-code trend.

ACKNOWLEDGEMENTS

We would like to express our gratitude to the participants at EduC for their support and input throughout the project. We are grateful to the reviewers of this paper for their valuable input in refining the article's content and providing more clarity where needed.

REFERENCES

[1] **Zhu, X., Ge, S. & Wang, N.** 2021. Digital transformation: A systematic literature review. *Computers & Industrial Engineering*, 162. Available from: https://www.sciencedirect.com/science/article/pii/S0360835221006781

[2] **Waqas, M., Ali, Z., Sánchez-Gordón, M. & Kristiansen, M.** 2024. Using LowCode and NoCode Tools in DevOps: A Multivocal Literature Review, in *New Perspectives in Software Engineering*. Mejía, J., et al. (eds). Cham: Springer Nature Switzerland, pp. 71-87.

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

[4] **Sahay, A., Indamutsa, A., Ruscio, D.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)*. pp. 171-178.

[5] **Frank, U., Maier, P. & Bock, A.** 2021. *Low code platforms: Promises, concepts and prospects. A comparative study of ten systems*. ICB-Research Report, Universität Duisburg-Essen, Essen.

[6] **Rashid, N. & Khan, S.U.** 2018. Agile practices for global software development vendors in the development of green and sustainable software. *Journal of Software: Evolution and Process*, 30(10), pp. e1964. Available from: https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.1964

[7] **Nerur, S. & Balijepally, V.** 2007. Theoretical reflections on agile development methodologies. *Communications of the ACM*, 50(3), pp. 79-83.

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

[9] **Cuellar, R.** 2012. Agile software development, in *The Next Wave of Technologies: Opportunities from Chaos*. Simon, P. (ed). Hoboken, New Jersey: Wiley, pp. 243-264.

[10] **Mahmoud, S.S. & Ahmad, I.** 2013. A green model for sustainable software engineering. *International Journal of Software Engineering and Its Applications*, 7(4), pp. 55-74.

[11] **Dick, M., Drangmeister, J., Kern, E. & Naumann, S.** 2013. Green software engineering with agile methods, in *2013 2nd international workshop on green and sustainable software (GREENS)*. IEEE, pp. 78-85.

[12] **Lebens, M. & Finnegan, R.** 2021. Using a low code development environment to teach the agile methodology, in *International Conference on Agile Software Development*. Gregory, P. (ed). Cham: Springer, pp. 191-199.

[13] **Lebens, M.C.** 2022. Student Perceptions of Low-Code Prototyping: A Case Study, in *International Conference on Life Sciences, Engineering and Technology*. Los Angeles: ISTES Organization, pp. 58-66.

[14] **Wong, J., Iijima, K., Leow, A., Jain, A. & Vincent, P.** 2021. Magic quadrant for enterprise low-code application platforms. *Gartner*, ID G00361584.

[15] **Reijers, H.A. & van der Aalst, W.M.P.** 2005. The effectiveness of workflow management systems: Predictions and lessons learned. *International Journal of Information Management*, 25(5), pp. 458-472. Available from: https://www.sciencedirect.com/science/article/pii/S0268401205000423

[16] **Käss, S., Strahringer, S. & Westner, M.** 2023. Practitioners' perceptions on the adoption of low code development platforms. *IEEE Access*, 11, pp. 29009-29034.

[17] **De Jong, P.** 2006. Going with the Flow: Workflow systems can provide value beyond automating business processes. *Queue*, 4(2), pp. 24-32.

[18] **George, A., Ali, M. & Papakostas, N.** 2021. Utilising robotic process automation technologies for streamlining the additive manufacturing design workflow. *CIRP Annals - Manufacturing Technology*, 70(1), pp. 119-122.

[19] **Gürcan, F. & Taentzer, G.** 2021. Using Microsoft PowerApps, Mendix and OutSystems in two development scenarios: an experience report, in *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)* IEEE, pp 67-72.

[20] **Mendoza Colom, C.** 2023. *Low-Code Technologies for Application Development*. Final Degree Project, Universitat Pompeu Fabra, Barcelona.

[21] **Vincent, P., *et al*.** 2021. Identify and Evaluate Your Next Low-Code Development Technologies. *Gartner*, ID G00742070.

[22] **Guilmette, A.** 2022. *Workflow Automation with Microsoft Power Automate: Achieve digital transformation through business automation with minimal coding*, 2nd ed. Birmingham: Packt Publishing Ltd.

[23] **Kirchhof, J.C., Jansen, N., Rumpe, B. & Wortmann, A.** 2023. Navigating the Low-Code Landscape: A Comparison of Development Platforms, in *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)* IEEE, pp. 854-862.

[24] **Al-Qassab, B.** 2021. *Ticket and worktime management system on Microsoft PowerApps and Common Data Service*. Bachelor's thesis, Häme University of Applied Sciences, Hämeenlinna, Finland.

[25] **Ciucan-Rusu, L., Timus, M., Stefan, D., Comes, C.-A., Bunduchi, E. & Poptamas, M.-A.** 2023. Low-code solutions integration for digital process automation-application in management, in *2023 22nd RoEduNet Conference: Networking in Education and Research (RoEduNet)*. IEEE, pp. 1-5.

[26] **Davlatov, N.** 2023. *Robotic Process Automation: Optimization of Business Processes with a Cost-effective Automation Tool*. Bachelor's thesis, Metropolia University of Applied Sciences, Kokkola, Finland.

[27] **Pandit, A.** 2024. *Exploring low-code and no-code development with Powerapps*. Bachelor's thesis, Centria University of Applied Sciences, Kokkola, Finland.

[28] **Vartiainen, H.** 2024. *Lean Management Empowerment: Elevating Employee Experience through RPA Implementation with Microsoft Power Automate*. Masters' thesis, Haag-Helia University of Applied Sciences, Helsinki, Finland.

[29] **Oates, B.J., Griffiths, M. & McLean, R.** 2022. *Researching information systems and computing*, 2nd ed. London: Sage.

[30] **Peffers, K., Tuunanen, T., Rothenberger, M. & Chatterjee, S.** 2007. A design science research methodology for information systems research. *Journal of Management Information systems*, 24(3), pp. 45-77.

[31] **Peffers, K., Tuunanen, T. & Niehaves, B.** 2018. Design science research genres: Introduction to the special issue on exemplars and criteria for applicable design science research. *European Journal of Information Systems*, 27(2), pp. 129-139. Available from: https://doi.org/10.1080/0960085X.2018.1458066

[32] **Flewelling, P.** 2018. *The Agile Developer's Handbook: Get more value from your software development: get the best out of the Agile methodology*. Birmingham: Packt Publishing.

[33] **Camburn, B., *et al*.** 2017. Design prototyping methods: State of the art in strategies, techniques, and guidelines. *Design Science*, 3, pp. e13.

[34] **Object Management Group.** 2014. *Business Process Model and Notation (BPMN), Version 2.0.2*. Available from: https://www.omg.org/spec/BPMN [accessed 31 August 2023].

[35] **Chinosi, M. & Trombetta, A.** 2012. BPMN: An introduction to the standard. *Computer Standards & Interfaces*, 34(1), pp. 124-134. Available from: https://www.sciencedirect.com/science/article/pii/S0920548911000766

[36] **von Rosing, M., White, S., Cummins, F. & de Man, H.** 2015. Business Process Model and Notation-BPMN, in *The complete business process handbook: Body of knowledge from process modeling to BPM Volume 1*. Van Rosing, M., Von Scheel, H., and Scheer, A. (eds). Amsterdam: Elsevier, pp. 433-457.

[37] **Leffingwell, D.** 2011. *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*. Upper Saddle River, New Jersey: Addison-Wesley.

[38] **Pacheco, C., García, I. & Reyes, M.** 2018. Requirements elicitation techniques: a systematic literature review based on the maturity of the techniques. *IET Software*, 12(4), pp. 365-378.

[39] **Tariq, S., Ibrahim, A., Usama, A. & Shahbaz, M.S.** 2021. An Overview of Requirements Elicitation Techniques in Software Engineering with a focus on Agile Development, in *2021 4th International Conference on Computing & Information Sciences (ICCIS)*. IEEE, pp. 1-6.

[40] **Ameller, D., *et al*.** 2019. Dealing with non-functional requirements in model-driven development: A survey. *IEEE Transactions on Software Engineering*, 47(4), pp. 818-835.

[41] **Fatoni, A., Adi, K. & Widodo, A.P.** 2020. PIECES Framework and Importance Performance Analysis Method to Evaluate the Implementation of Information Systems. *E3S Web of Conferences*, 202, Article 15007, pp. 1-11.

[42] **Muslih, M., Wardhiyana, L. & Widianto, S.R.** 2021. Analysis and Evaluation of ERP Information System User Satisfaction PT. Bozzetto Indonesia Using Pieces Framework. *Jurnal Mantik*, 4(4), pp. 2588-2598.

[43] **Khan, J.A., Rehman, I.U., Khan, Y.H., Khan, I.J. & Rashid, S.** 2015. Comparison of requirement prioritization techniques to find best prioritization technique. *International Journal of Modern Education and Computer Science*, 7(11), pp. 53-59.

[44] **Paul, D. & Cadle, J.** 2020. *Business analysis*, 4th ed. Swindon, UK: BCS, The Chartered Institute for IT.

[45] **Vestola, M.** 2010. A comparison of nine basic techniques for requirements prioritization. *Helsinki University of Technology*, pp. 1-8.
[46] **Kralik, L., Jasek, R., Zacek, P. & Senkerik, R.** 2019. Agile approach in multi-criterial decision making. *International Journal of Manufacturing Technology and Management*, 33(3-4), pp. 256-267.
[47] **Ramayasa, I.P. & Candrawibawa, I.G.A.** 2021. Usability evaluation of lecturer information systems using Sirius tramework and MoSCoW technique. *Scientific Journal of Informatics*, 8(1), pp. 16-23.
[48] **Microsoft Power Platform.** 2025. *Power Platform Licensing Guide August 2025*. Available from: https://cdn-dynmedia-1.microsoft.com/is/content/microsoftcorp/microsoft/bade/documents/products-and-services/en-us/bizapps/Power-Platform-Licensing-Guide-August-2025.pdf [accessed 3 August 2025].
[49] **Mercurio, R. & Merrill, B.** 2021. *Beginning Microsoft 365 collaboration apps: Working in the Microsoft coud*, 2nd ed. New York: Apress.
[50] **Ribeiro, J.B., Amorim, M. & Teixeira, L.** 2022. E-roadmap to Support the Digital Transformation: Study, Design and Development of a Tool to Accelerate Digitalization in Lean Companies, in *Proceedings of the 5thEuropean International Conference on Industrial Engineering and Operations Management*, Rome: IEOM.
[51] **Dorrani, R.** n.d. *How to save Microsoft Forms responses and attachments to SharePoint lists or libraries and send email*. Available from: www.youtube.com/watch?v=K-hiDOPAG-4 [accessed 24 November 2024].
[52] **Jones, G.** 2020. *Microsoft Teams approval workflow using Power Automate - part 1: Microsoft Teams Tutorial 2020*. Available from: www.youtube.com/watch?v=DkEtgdM17UQ [accessed 24 November 2024].
[53] **Jones, G.** 2020. *Microsoft Teams approval workflow using Power Automate - part 2: Microsoft Teams Tutorial 2020*. Available from: www.youtube.com/watch?v=nkFCKW-IN8A [accessed 24 November 2024].
[54] **Jones, G.** 2020. *Microsoft Teams approval workflow using Power Automate - part 3: Microsoft Teams Tutorial 2020*. Available from: www.youtube.com/watch?v=NhWLOUOfn_k [accessed 24 November 2024].
[55] **Microsoft.** 2023. *Microsoft Power Automate Community Forum*. Available from: https://powerusers.microsoft.com/ [accessed 30 November 2023].
[56] **Evans, W. & Petersson, B.** 2023. *How does low-code development correspond with best practice in software development?* Bachelor's thesis, Malmö University, Malmö, Sweden.
[57] **Palmer, T.** 2020. *Microsoft PowerApps as an Alternative Solution to Business Application Development*. Bachelor's thesis, Haaga-Helia University of Applied Sciences, Finland.
[58] **Ferdous, J.** 2025. *Using Microsoft Power Platform to automate daily redundant tasks in an organization*. Bachelor's thesis, Tampere University of Applied Sciences, Finland.
[59] **Miscrosoft.** 2025. *Microsoft Ignite: Drafts and versioning for solution-aware cloud flows*. Available from: https://learn.microsoft.com/en-us/power-automate/drafts-versioning [accessed 3 August 2025].
[60] **Burke, C. & McArthur, S.** 2024. *Introducing Git Integration in Power Platform*. Available from: https://www.microsoft.com/en-us/power-platform/blog/power-apps/introducing-git-integration-in-power-platform-preview/ [accessed 3 August 2025].
[61] **Abelita, K.** 2023. *Analysing the Efficiency of the Software Development Process in a Low-Code Technology Company: A Case Study Using Value Stream Map*. Bachelor's thesis, Kerelia University of Applied Sciences, Joensuu, Finland.
[62] **Lichtenthäler, R., Böhm, S., Manner, J. & Winzinger, S.** 2022. A Use Case-based Investigation of Low-Code Development Platforms, in *Proceedings of the 14th ZEUS Workshop on Services and their Composition*. Bamberg, Germany, pp. 76-83.
[63] **Käss, S., Strahringer, S. & Westner, M.** 2023. A Multiple Mini Case Study on the Adoption of Low Code Development Platforms in Work Systems. *IEEE Access*, 11, pp. 118762-118786.