Selected Papers of the IRIS, Issue Nr 15 (2024) Scandinavian (IRIS)

8-2024

LOW CODE DEVELOPMENT: PROFESSIONALS, TASK-FORCES, AND THE WILD WEST

Marcus B. Engelsen

Jacob Nørbjerg

Jeffry Babb

LOW CODE DEVELOPMENT: PROFESSIONALS, TASK-FORCES, AND THE WILD WEST.

Research paper

Engelsen, Marcus B., Deloitte, Copenhagen, Denmark, M.B.Larsen@outlook.dk

Nørbjerg, Jacob, Copenhagen Business School, Copenhagen, Denmark, jno.digi@cbs.dk

Babb, Jeffry, West Texas A&M University, Canyon, TX, USA, jbabb@wtamu.edu

Abstract

Low Code Development (LCD) platforms from IBM, Microsoft and other vendors have reinvigorated end-user development. Business professionals without extensive skills or background in systems and software development are empowered to create new IT and software artefacts through the extended capabilities of these platforms. The business professionals developing IT solutions with the help of the LCD platforms are usually referred to as 'citizen developers', but in practice, from an IT governance standpoint, these citizen developers are also referred to as "cowboys" as they leverage the LCD platforms in the hinterlands and frontier of IT Governance's comprehension and reach. Whereas the idea of end-user development is not new or novel, the low-code context presents a fresh occasion to understand the breadth and depth of IT professionalizing and present implications for IT Governance, training, and the low-code platforms themselves. This paper theorizes on the emerging challenges and opportunities inherent in the conceptualization of the citizen developer in low code/no code environments.

Keywords: Low code/no code, RPA, IT Governance, IT Professionalization.

1 Introduction

Low-code development (LCD)[1] platforms and tools enable development of powerful software solutions with few – if any – lines of traditional program code, by using powerful commands and visual programming tools with drag and drop capability. LCD is expected to increase reduce implementation delays and improve software development productivity and innovative capabilities (Binzer and Winkler, 2022, Elshan et al., 2023). The intuitive interface offered by the LCD platforms, furthermore, enable organizations to allow business professionals with little or no programming experience to develop and deploy business applications without help from IT professionals – also known as 'citizen development' (Oltrogge et al, 2018; Binzer & Winkler, 2022).

Organizations use citizen development to circumvent the IT function's control over IT development, overcome the IT resource bottleneck and support business agility and innovation by providing quick solutions to evolving business needs (Elshan, 2023; Osmundsen et al., 2023). However, there is a tension between rapid and decentralized citizen development and organizational oversight and governance (Osmundsen et al., 2023). LCD can create problems such as, e.g., local process optimization, siloed applications, poor software quality, issues with integration with core IT systems, wildly growing and poorly coordinated applications, future maintenance challenges, etc. There is, therefore, an ongoing debate about how to balance decentral and central governance mechanisms and structures to coordinate development and deployment of applications without sacrificing decentral innovations and short development cycles, as well as train citizen developers in proper use of the LCD platform. (Osmundsen et al. 2023; Noppen et al., 2020; Willcocks et al. 2018).

Powerful software tools that increase ISD productivity and enable business professionals to build applications is not a new phenomenon. The computer and software industries have always worked to improve programmer productivity, systems quality, portability etc. by extending the basic functionality of the computer – the computer systems core (Friedman, 1989) – with more advanced development tools, i.e., replacing binary machine code with assembly languages, followed by 3rd and 4th generation programming languages, formal specification tools, database management platforms, and software libraries (Friedman, 1989, Pinho et al., 2023). These inventions were accompanied by repeated predictions of the demise of programmers and IT departments, and the rise of end-user or business professional development (Friedman, 1989). Experiences were mixed, however. The tools were not as powerful and intuitive in use as claimed, and business developers needed more training than expected to build anything but simple applications. The IT departments furthermore had to step in and resolve quality and other issues with the applications (Carlsson, 1989; Friedman, 1989; Heijmans, 2015; Lethbridge, 2021; Ngwenyama, 1993).

Although sharing many of the characteristics of previous waves of end-user development, contemporary citizen development also exhibits new potential: Increasing hardware performance enables more powerful tools and intuitive graphical drag-and-drop interfaces. Modern LCD platforms also include better support for professional software engineering practices than earlier 4GL tools (Heijmans, 2016). This, together with a general increase in IT proficiency among business professionals, enable the citizen developer to undertake more complex development tasks than in previous waves of end user computing.

Current research into LCD focuses on general benefits and challenges of the technology and citizen development, as well as how to balance governance and creativity in citizen development (Osmundsen et al., 2023; Wilcocks et al., 2015). Our research focuses first and foremost on the actual LCD practices in a large multinational consultancy. Our investigation reveals different LCD scenarios, with different development and management practices, ranging from tightly managed contract development – for external and internal customers – by professional software developers to unregulated citizen development for personal or local team purposes.

In this paper, we describe outcomes from embedded qualitative research that examines the implementation and use of LCD platforms where subject matter experts were introduced to LCD tools used to develop applications which utilized a variety of data integrations and connections. These 'citizen developed' LCD artefacts quickly took on their own vitality and utility for the SMEs in ways that were not

fully comprehended by the IT governance function. This informs the central research question for the paper:

RQ: How can organizations utilizing LCD integrate citizen developers into their broader IT governance and professional paradigms?

Evidence from the study suggests that the unregulated nature of effective and motivated citizen developers is a byproduct of the full effect of the innovation coming to bear. As such, it is conceivable that the governance and professional development posture of the organization must be prepared to approach the citizen developer as being both a business professional and a software developer, and elevate these citizen developers in terms of acculturation, support, and professionalization.

2 Related research

In the following we summarize the benefits and challenges reported in recent research about citizen development with LCD. We conclude the section with a discussion of how to manage the tension between the creativity and user-driven rapid solution development enabled by LCD, and the long-term risks for the organization caused by the lack of governance of a multitude of applications built by business professionals without basic knowledge of sound software engineering standards and techniques. This discussion leads to the formulation of our research question.

2.1 Benefits of LCD

Experience from LCD development projects confirm the potential to accelerate development and decrease dependency on tech skills, thereby reduce the IT backlog and dependence on the IT function. Martins et al. (2020) report how an HR self-service portal was developed in three days with the platform *Outsystems.* Woo (2020) similarly reports how New York City officials built an online portal for collecting information from COVID-19 patients in just three days without a single line of code.

Research also confirm that LCD allows business professionals to create their own IT solutions to problems in their daily work. This helps close the gap between IT and business skills and unleash the creativity and innovative capabilities of business professionals (Binzer and Winkler, 2022; Elshan et al., 2023; Li et al., 2022) as well as improve IT-business alignment (da Silva Costa et al., 2022).

The powerful and intuitive LCD platforms support short (agile) development-test cycles and close collaboration between the involved parties (Binzer and Winkler, 2022).

2.2 Challenges of LCD

The challenges with LCD and citizen development arise from factors related to the platforms, the limited technical and software engineering knowledge of the citizen developer, and poor organizational oversight and control of the installed LCD applications (cf., Eulerich et al., 2024; Khorram, 2020; Lethbridge, 2021, Willcocks et al., 2018). Oltrogge et al. (2018) report that 11% of free apps developed with LCD platforms had major security risks such as poor access right control and vulnerability against http attacks. Errors and security risks are amplified when auto-generated code is used in multiple applications unless the application developers have the skills needed to identify and remove the vulnerabilities.

Sahay et al. (2020) evaluated eight low-code development platforms and found four key challenges: (1) a high risk of vendor lock-in due to poor interoperability across platforms; (2) poor extensibility because most platforms do not allow for the addition of new functionality; (3) a steep learning curve, which inhibits adoption of some platforms; and (4) to include more users and functionality.

The limited IT skills of the citizen developers create other risks. Khorram et al. (2020) reports that the limited technical knowledge and software skills of the citizen developer results in little or insufficient quality control, particularly with regards to testing. The LCD platforms offer high- level test automation in the form of data-driven", model-based, and record-and-replay testing, but they still require some manual scripting, which does not align with the low-code development techniques and the skills of the citizen

developers. Furthermore, there is a need for testing cloud-based applications and not many tools exist for automatic testing in the cloud (Khorram et al., 2020).

2.3 IT governance in the LCD Context

At its core, IT governance is a risk appraisal and mitigation venture to assure the efficacy and availability of an organization's IT function. Among the risks to be understood and mitigated are the effects of innovation and growth. When an innovation becomes available which allows developers to connect to productivity, insight, and advancement, it is often the case that any downstream issues from an IT governance standpoint are latent and largely undetected. With LCD environments, that opportunity to capitalize on advancements in throughput is the selling point of the platforms. However, it is also the case that uncontrolled growth of LCD apps, i.e., citizen development, may lead to poorly designed high-code apps over time (Lethbridge, 2021) As such, initial gains from accelerated progress accorded to LCD may be eroded over time once the low hanging fruit of innovation introduction have been harvested (Eulerich, 2024; Osmundsen et al. 2023; Wilcocks et al., 2018). Furthermore, it is likely that only a handful of citizen developers will create truly impactful breakthroughs which exist among a large volume of citizen-developed apps in the environment. This creates a potential for both knowledge management and skills issue. Willcocks et al. (2018) warns against (among others) *skills shortages* when LCD experts and application owners leave the company, *duplicated applications*, and *integration challenges* among LCD applications and between LCD applications and other systems.

These issues can be understood as classic IT governance challenges where the tensions of control, innovation, productivity, and insight must be understood, weighed, balanced and managed (Osmundsen et al.., 2019). This becomes a matter of the degree of coupling the innovation to governance controls. Models of the value of the IT function, how LCD fits within that function, and how innovations produced within the LCD environment are needed. This should lead to insights that transcend governance and lead to questions and inquiry as to the professional nature of the citizen developer in the broader context of the organization's IT function.

Whereas some researchers agree on the potential, benefits, and challenges of LCD driven by business professionals, there is a lack of studies into alternative LCD development scenarios and models of their professionalization in the IT context (Binzer and Winkler, 2022). Studies of other potential use cases, i.e., LCD by IT professionals or teams of IT and business professionals are hard to find.

The results of the study conducted to support this paper explores these alternative models and inform our research question. We describe and characterize three possible scenarios and propose a fourth to integrate both the on-boarding of the citizen developer into the IT realm and to adjust concepts and models for IT governance that recognize and accommodate the citizen develop as an IT professional.

3 Case description

This study is based on an embedded qualitative research setting involving a professional services company that employs more than 300.000 people across more than 150 countries and with an annual revenue of more than 45 billion US dollars.

The organization has a central headquarters that hosts most executive functions and additional regional officers. The headquarter (The Executive) entails the global leadership, governance bodies and three geographic Areas – Americas; Europe, Middle East, India and Africa (EMEIA); and Asia-Pacific. There are 28 separate regions grouped within the overall areas that deliver professional services to clients through projects, temporary resource allocations or advisory services. The regions are further organized in four client serving business units: Assurance, Tax, Consulting, and Strategy and Transactions, and one internal unit Core Business Services.

In 2018 the company transitioned and upgraded their Microsoft Office 365 suite to include the Power Platform. The Power Platform entails Power BI for data visualization, Power Apps for application building, Power Automate for workflow automation and Power Virtual Agents for chatbot configuration. The

upgrade is available to every employee in the company and costs hundreds of millions of dollars every year in licenses.

This transition and the LCD provisions within the platform, is the subject of the study. Whereas the initial interest of the study was the IT governance issues related to this transition, the embedded nature of the research soon revealed additional issues that inform the research question in this paper that relates to the conundrum presented by the 'citizen developer'.

4 Research method

We used a single embedded case study across multiple departments and layers in the organization, providing a holistic view of the phenomenon (Yin, 2003). One of the authors engaged in a series of interviews and observations, using a predominantly inductive approach to data collection and analysis to identify and refine emergent patterns (Creswell, 1994). By letting the empirical data guide the direction of the study, the materialization of theoretical perspectives related to LCD were developed in a manner that is consistent with an interpretivist methodology.

The interviews were conducted digitally following semi-structured interview guides ensuring the overall direction of the interviews, while allowing the respondents to bring their own worldview into their answers. The study lasted four months going from February-June 2021 and involved 10 use case interviews and four governance interviews.

Name and function	Tenure	Department and location	Background
Tony, Delivering robotics solutions to clients	3 years	Robotics and Artificial intelligence EMEIA, Johannesburg	Software engineering, business systems and analytics
Pete, Client serving role focus on data analytics	12 years	Strategy and Transactions – Transaction diligence, EMEIA – Switzerland	Business and economics
Ben, Building reports and data visualizations	7 months	Core Business Services	Biotechnology and chemistry. Data and analytics
Patrick, Designing and developing solutions	9 months	Core Business Services, EMEIA Denmark	Business and IT
Bill, Managing IAM infrastructure	2+ years	InfoSec – Active Directory Infrastructure	Computer engineering
Rajiv, Preparing indirect tax returns for clients	2+ years	TAX – Indirect tax	Commerce background
Fred, Implement the Power Platform with clients	4 years	TAX – Tax technology and transformation	Accounting
Somnath, Implement old process into new process w. Power Platform	2+ years	Core Business Services – Project process implementations	Computer engineering, Master in information technology
Vicki, QA of consulting services	3 years	Asia Pacific Consulting quality team, Australia	International business, Project management
Adam, Delivering development solutions to clients	2+ years	Intelligence service, Peru	Business

Table 1 Overview of the LCD respondents and their place in the organization

Respondents representing use cases were selected through self-selection sampling by posting a request for participation on internal LCD community sites. Based on 20 received replies, purposive heterogeneous sampling was applied in selecting the final 10 respondents. The final 10 respondents were selected based on a decision to narrow the field of study to focus on a single LCD platform, where all 10 had used Microsoft Power Platform in the past.

Table 1 is an overview of the use case respondents and their place in the organization. These individuals were those who were actively involved in LCD. The names are fictitious to preserve the anonymity of the participants.

Additional respondents were selected to better understand the governance aspect of the study. See table 2. These respondents were selected through snowball sampling, where one respondent was first identified and interviewed in which more respondents were identified. The first respondent was interviewed twice as new information came to light during the interview process.

Name and function	Tenure	Department and location	Background
Joshua, Looking after the Power Platform	2-3 years	Technology – End User technology Experience	General IT
Michelle, Assisting with building internal solutions on top of Office 365 dynamics or Power Platform	2 years	Client technology – low-code services	Microsoft technologies
Toby, Review and assess risks on internal platforms	2+ years	Security consulting end-user computing	Software engineering experience (infrastructure)

Table 2 Overview of the governance respondents and their place in the organization

The interviews were analyzed following the thematic analysis framework by Braun & Clarke (2006). 400+ codes were discovered across all 14 interviews, which were then grouped in four overall themes: "Internal formal use", "Internal informal use", "External formal use" and miscellaneous. The discovered themes, with exception of the miscellaneous theme, provided key insight into the phenomenon in scope of the study and defined the direction of the further analysis. Offsetting from the three main themes, theoretical perspectives from software engineering best practice were included to organize the insights according to a common framework.

5 Analysis

We have structured the analysis according to the organizational scope (external vs. internal), and how development is managed (formal vs. informal) of the LCD projects: The external-formal teams developed applications for external customers based on contracts. The internal-formal team produced solutions for internal customers based on a formal agreement, and finally, the internal-informal team made solutions for themselves or their immediate co-workers on an ad-hoc basis.

5.1 External-formal

These developers work in two different client serving teams. They use the Power Platform to increase sales to external customers:

> *"We're trying to make this new service focusing intelligence services. We're not trying to sell low-code because low-code is a concept"* (Adam)
>
> *"I've kind of been working on the, you know directly with clients on how to implement the Power Platform for different processes that they have"* (Fred).

The projects vary between, e.g., implementing a COVID care application for a farming company in Peru (Adam), to building a Power Platform Center of Excellence (CoE) with the African clients (Tony).

The way these solutions are delivered vary between client implementations and managed service:

> *"we're just onboarding client users onto our environment so that is like 80% of the use case. 20%, we build a product. And we deploy to the client's environment"* (Michelle).

Getting the client's sign-off is a part of the company's internal processes to initiate an engagement with a client, and the developers focus on gathering proper requirements prior to solution design.

> *"[Requirements] (...) can be a bit fixed purely because we require that client sign off because we need to manage scope that way (...) what will normally do is we'll have an overarching waterfall approach, but every single requirement that we develop until later on in the build phase will be done very iteratively"* (Tony).

The design of the solution begins with a design sprint focused on understanding the requirements more in detail along with the required infrastructure and resources (Tony). After this sprint the teams have a clear understanding of what needs to be developed. Based on those requirements the teams will start development sprints.

> *"(...) we have the other sprints, development [and] quality of the release one (...) development and quality of release two"* (Adam)
>
> *"(...) we'll go into build and pretty much every two weeks we'll try and release (...) and we'll have generally a feedback workshop as well with our stakeholders (...)"* (Tony).

Quality assurance is after the completion of a sprint prior to beginning the next sprint. The approaches of the two teams are very similar and contain various types of testing. One approach they both utilize is testing with the individual developer:

> *"(...) the developer will do their unit testing in the development environment (...)"* (Tony)
>
> *"(...) the developer is responsible to code and test (...) the component"* (Adam).

Tony's team also apply beta testing, allowing a small group of users to access the solution:

> *"(...) we'll test it out, put it into what we call a soft go-live state where we get some beta testers in"* (Tony).

Adam's team utilize an actual tester rather than the users to test the code before deployment.:

> *"(...) and after that [developer testing] we have integration integral testing with the tester, it's on is another role"* (Adam).

Both teams use acceptance testing prior to final release of the completed product.

The teams offer to maintain and support a client solution after deployment for a limited time period, however this requires a formal agreement (Adam; Tony).

5.2 Internal-formal

The internal-formal teams use Power Platform to optimize, digitize and automate internal processes in the broader organization. The overarching goal is cost reduction.

> *"So this team I'm working in (...) has the purpose of automating digitizing [the company] from the inside (...)"* (Patrick)
>
> *"Project process implementations (...) we implement old process into the new process and for the Internal stakeholder"* (Somnath).

The projects vary, but they usually revolve around an internal pain point such as *"every year there is a whole process in place which was done manually (...)"* (Patrick). Both teams develop solutions involving Power Apps as a user frontend, Power Automate as backend and Power BI for reporting (Tim; Somnath).

Tim's team follows a structured approach to requirements gathering, walking through the existing process with business stakeholders.

> *"(...) the business stakeholders and (...) the project manager were already in discussions before (...) But once I joined, we tried to take it from the beginning again and go through the process"* (Patrick).

The project process implementation team also had the project manager gather requirements

> *"(...) project manager gather all the requirements from the team and then he discussed with me everything"* (Somnath).

Tim mentions producing documentation in the early stages of the project: *"(...) process diagram (...) solution architecture (...) user stories"* (Patrick). These documents help Tim's team gain a deeper understanding of what needs to be developed. The client would accept on the POC, acting as a sort of informal sign off (Patrick).

There are many changes to the requirements during construction despite the initial agreement sign-off, and the team spend little time on documentation during construction. They prefer to *"build it and move on to the next thing as fast as possible (...)"* (Patrick).

Tim's team used to work in sprints during construction with daily stand-up meetings to monitor progress, and using user stories as requirements (Patrick). The constant changes affected the team's use of sprints, however:

> *"(..) we kind of lost track of some of the items and (...) didn't really plan and move in sprints anymore (...) We constantly were showing it to people and (...) we still had weekly calls with the business stakeholders in order to get their input (...)"* (Patrick)

Somnaths team has one developer working on the requirements managed by the project manager, and no formal structure for the actual development (Somnath).

The follow the unit-systems-acceptance test process. Once all modules have been tested and the solution ready to go live, the users were asked to test

> *"(...) we had a small group (…) six individuals and what we tried to get some more feedback from them (...) like when the big crowd comes that everything is already fully in place"* (Patrick).

Tim's team continue working on the solution after deployment.

> *"So while we were live I was still developing (…) stuff further. Like that, suddenly another type of user had to be able to see the same information, so I had to build a new screen for them (...)"* (Patrick).

The use of the Power Platform to optimize and digitize the company follows a partly structured approach. Initial requirements are discussed before the project begins, but evolve and change during construction – often without being documented.

Some quality assurance is made on the solution, but constant changes from the internal clients challenges tracking of systems changes and testing. There is no formal handover from development to use, and the responsibility for further maintenance stays with the developers.

5.3 Internal-informal

In the internal-informal scenario business professionals, i.e., citizen developers, in the entire company use the Power Platform to optimize their own or the work of their team.

> *"I've actually started using it just to automate my own work a lot of the time (...)"* (Tony)
>
> *"(...) I started to use power automate to create different types of flow for my team (...)"* (Somnath).

The citizen developers want to relieve themselves and/or their team from inconveniences, e.g., lack of overview, or reduce manual work. Examples include *"(...) power BI reports [about] the status of your engagement"* (Fred), or *"(...) a form submission and then using power automate to email the end results (...)"* (Ben).

The projects are self initiated and informal

> *"(...) I want to automate that particular process, so (...) I went to the SharePoint team for a solution and they said that is not possible with them (...)"* (Bill).

There are no formal requirements gathering since the citizen developers already know the process they want to improve.

> *"(...) I knew exactly what I wanted to achieve (...)"* (Pete)
>
> *"I just kind of, you know, knew (...) what the process was, and what I needed to put in place"* (Vicki).

Team focused projects are sometimes initiated by a team member. They are very similar to the entirely self-initiated projects with the exception that they will be discussed with a manager.

> *"I actually went to her and kind of pitched this idea (...)"* (Fred).

Some requirements may be gathered in team oriented projects

> *"(...) a lot of it was evolved from actually liaising with the actual end users and making sure that their you know their feedback was incorporated"* (Ben)
>
> *"my team if I say they are the actual users of this solution, so I discussed with them and I collected their inputs (...)"* (Bill).

Both individual and team oriented projects involve the end-user(s) continuously during development through a continuous feedback loop rather than actual requirement specification.

The citizen development process in the internal-informal scenario continues the informal pattern from the requirements gathering

> *"I wouldn't say there was any [process]. I guess formal structure that we followed like any sort of, you know, like you know best practices and stuff like that (...)"* (Ben)
>
> *"(...) just started playing around with it, I guess"* (Vicki).

A few of the respondents mention available tools in the company when developing the solution:

> *"(...) I've never had proper infrastructure in the sense of like SQL servers or anything like that always was a bit restricted with what's available for free in the company (...)"* (Pete)

For the actual implementation internal-informal projects follow an informal iterative process:

> *"(...) so we had sometimes iterations where, let's say the account role app. We first include the certain roles and then it turned out that teams maintain more roles themselves in Excel (...)"* (Pete).

There is no structured testing process or plan. For end user testing a citizen developer would just reach out and ask colleagues to try it out.

> *"(...) the only important thing was to make sure that it works for everybody who for whom it needs to work and there were a couple of times I had to ask other people to try it for me"* (Pete).

The citizen developer (or the team) retains responsibility for ongoing maintenance and development.

> *"(...) it's kind of on me to. Just make sure everything is working (...)"* (Fred)
>
> *"They are maintained by me or I have now some people in our offshore centers people who are sort of partially familiar with the setup"* (Pete).

Time to maintain and further develop a solution is thus taken from other business. This seems to be of little concern to the citizen developers:

> *"[It] just didn't seem to break at all"* (Ben)
>
> *"(...) I never notice any fails so far. It is running more than a year now"* (Bill)
>
> *"(...) there's some legacy flows that I still have running that had just worked for two years now (...)"* (Fred).

New features are continuously added to the solutions based on informal user feedback. This is a general pattern followed in all the cases that deploy a version of the solution to the users during testing.

> *"during testing we've already got some feedback on potential improvements"* (Vicki).

There is no documentation produced for the original solution or subsequent changes (Vicki; Ben; Bill; Rajiv).

5.4 Governance

The company encourages the general use of the Power Platform as explained by the product manager for the platform

> *"(…) to try and, you know, get our money back on the investment (…) my success be measured by how much I increase adoption (…)"* (Joshua)

Enterprise-wide governance mechanisms are in place, however, to ensure proper information security and control IT risks. These mechanisms include different levels of access to the Power Platform's functions depending on the scope of the projects and the developers' capabilities.

> *"[W] have the default environment which we currently use as personal automation (…) the non-default environments (…) to a more enterprise type solution building (…) then we have the dedicated environments which are used for like those niche (…) solutions"* (Toby)

5.5 Analysis summary

The three scenarios represent stereotypical approaches to LCD, which we metaphorically label the Professionals, the Taskforce, and the Wild West. See table 3.

5.5.1 The professionals

The Professionals are so named as their approach overall is very structured and successfully follow professional software engineering practices. They use LCD to increase sales by delivering good software quality faster to the clients. Starting with a formal analysis and requirements specification phase the professionals formally agree with an external client on the scope to be delivered, including how to deliver and to what extent the client is involved. Some development approaches follow a structured hybrid of agile and waterfall development, delivering on the overall scope through iterative construction beginning with an MVP. Software validation is conducted using well-known test practices and involves multiple types of testing according to the formally agreed approach. Responsibility for the further evolution of the solution is mostly left with the client, as this usually exceeds the scope of engagement, although agreements for operation and further development can be made with the client.

The activities of the professionals are externally regulated by formal client contracts, and internally by their adherence to software engineering practices.

The professionals have full access to the Power Platform.

5.5.2 The Taskforce

The Taskforce have gotten their name as their approach is structured but focused more on results rather than the process of getting to the results. The overall goal of the Taskforce is focused on using LCD for internal cost optimization through digitalization. Similar to the professionals, the taskforce begins by gathering requirements from the client, which in this case are internal units or functions in the organization. Following this, a design will be made and presented to the client with the aim of obtaining a formal – although not contractually binding – sign-off. Development and validation initially follow a structured process, however, the structure is quickly abandoned when priorities change in the team or organization. As opposed to the professionals, who agree on the process with the client, the Taskforce define their own approach regarding development and validation activities, which usually involves ongoing dialogue and check-ins with the client. Following the development the Taskforce deploy and maintain the solution. Further evolution, i.e., implementation of new requirements or fixing minor bugs – is handled by the taskforce.

The projects undertaken by the Taskforce are governed by semi-formal agreements with the (internal) customers. Their practices are partly ad hoc and partly in accordance with common software engineering practices.

The Taskforce have full access to the Power Platform.

	External-formal use *The Professionals*	Internal-formal use *The Taskforce*	Internal informal use *The Cowboys*

Customers	External	Internal	The citizen developer and colleagues
Goal	Building solutions for external clients increasing sales	Digitizing and optimizing internal clients work reducing operational cost	Automating or optimizing own or team's work reducing inconvenience
Specification	Formally gathered requirements specified in a formal agreement getting client sign-off prior to development. Low flexibility	Formally gathered requirements specified in a requirements document requiring no official sign off. Evolving with limited documentation	Informally gathered requirements based on team meetings or own knowledge, with no documentation. Evolving.
Implementation	Work in sprints delivering increments to client and utilize first sprint to understand requirements.	Structured approach to work following either sprints or a plan and adapts to change throughout development.	No formal structure of the work, solutions is a constant work in progress. Ad hoc improvements.
Testing	Structured approach to testing utilizing three types of testing with user acceptance test marking the official handover to the client.	Seemingly structured approach to testing onboarding small user groups at the end of development prior to full scale.	Unstructured approach to testing utilizing the concept of trial and error in development efforts and early launch to teams when first version is ready.
Maintenance	Time limited maintenance formally agreed with client, addition of new functionality when agreed with client.	Maintenance kept with developer and team, taking on feedback and planning implementations based on feedback.	Maintenance kept with developer and team having little concern as stability is high and new functionality is added ad hoc.
Control mechanism	External 'contract' (limited flexibility) and formal accept	Internal explicit agreement. Flexible.	Informal. Ongoing negotiations (if multiple users).
Access to Power Platform features	Extended environment	Extended environment	Default environment
Developer profile	Software professional	Software professional	Business professional

Table 3 Different approaches summarized according to stereotype

5.5.3 The Wild West

The Wild West are so named as their approach to development is following pretty much whatever they want to do and how. Their overall goal is centered around increasing productivity by reducing time spent on mundane tasks for themselves or their team. In the Wild West project ideas originate from individuals or teams, and construction starts immediately after informal chats with colleagues. Construction is done on an ad-hoc basis interspersed with regular business tasks. The result is validated by the developer as well as through ongoing demonstrations to team-members. Having built the solution, the developers continue to be responsible for operation, maintenance and further development.

Projects are defined and executed by the business professionals themselves, and there are no external mechanisms in place to control or allocate development resources. Their construction and quality assurances processes are ad hoc. There is no central overview of operational solutions or their status.

The Wild West developers have access to the default version of Power the Power Platform.

5.5.4 A fourth stereotype?

The three scenarios and the associated practices differ on two dimensions: *Organizational scope* and *formality of agreement with customers/users*. The first dimension pertains to whether the scope of the

development is focused at delivering software externally or internally. The second dimension refers to whether the agreement on the delivery of software is formal or informal. The dimensions and stereotypes are illustrated in Table 4 below:

		Formality of agreement	
		Formal	Informal
Organizational scope	Internal	Taskforce	Cowboys
	External	Professionals	?

Table 4 Dimensions and Stereotypes

The two-by-two matrix captures the three stereotypes identified in this study, but implicitly hypothesizes a fourth stereotype: Informal-External. This fourth stereotype, although not identified in the study, could be found in situations where a consultant is allocated to a client over a longer period to deliver solutions using an LCD platform.

6 Discussion and conclusion

Previous research (e.g., Osmundsen et al., 2023, Wilcocks et al., 2015, 2018) focuses on governance structures and how to reconcile the innovative potential and short development cycles of citizen development with the traditional, more centralized IT governance. The latter – dominated by the traditional IT department – risking to drown innovation and problem solving in rules, regulations, and limited resources. This study adds a new dimension to the debate by identifying different LCD scenarios and actor stereotypes whose activities reveal new relationships between professional and novice developers. The Professionals and the Taskforce used LCD as a powerful development tool *within* standard software engineering and systems development practices, thus combining controlled development of quality software with the productivity gains provided by the tool.

In the LCD space, and within the affordance framework of the underlying platform, there is some convergence between the citizen developers in the Wild West, and the professional developers in the other scenarios. However, their training, perspectives, instincts, and design orientations are dissimilar. While the citizen and the professional developer have the powerful tool in common, they do not approach the tooling or the outcomes in the same way.

We do not foresee the feasibility of developing a unified model of IT governance that perfectly incorporates the LCD environment. While there is guidance in the literature on LCD implementations, including specific use cases, which highlight challenges, opportunities, and considerations in selecting and implementing LCD platforms, we have found only one example of research that explores LCD platforms use where the impacts on the involved actors are considered. Virta (2018) has investigated how low-code development relates to standard software development through a case with a Finish consultancy – Biit oy – and their use of Salesforce. She discovers that low-code tools are perceived to reduce development cycle time and offer simple process automation, however they also perceive them to be inefficient, have performance issues with large data volumes, provide maintenance issues with complex solutions and have an obscure nature to their offered tools. The primary utility of this case is how it highlights issues surrounding implementation, acculturation, and unintended/unanticipated issues of use.

Characterized as the Wild West the citizen developers – or to stretch the metaphor: the cowboys – in the third scenario clearly contributed to the overall value of the organization in their use of the new LCD

ecosystem. However, as they have been neither internally nor externally acculturated, indoctrinated, nor trained as IT professionals, they lack the instincts and orientation of the professional. As a matter of IT governance, this raises questions of responsibility and accountability as to where the imperative lies to incorporate the citizen developers. In many respects, the cowboys in the Wild West are not organizationally informal actors at all, but the evidence from the study reveals that they are regarded as being 'outside' the walls of IT governance. More research is needed to explore the relationship between professional skills, LCD practices, and governance. Maybe one size governance does not fit all.

Although at present we do not fully answer the research question, we do have the elements on hand to progress towards an answer. Among the available strategies is to 'deputize' the cowboys in the Wild West through education and training. More research is also needed in this regard.

As we have used stereotypes to understand the dynamics between the actors in the case, we can best describe a possible stereotype for the citizen developer by borrowing circumstances where other hierarchical organizations have identified an entity that occupies the 'in between' space. Many armed forces around the world have identified the need for a role known as a 'warrant officer'. In this case, these are roles where the service required is not fully that of a commissioned officer (the professionals from our case) but elevated above the responsibilities and scope of an enlisted soldier. As the structure goes, these individuals are lesser (in experience and responsibility) among officers, but senior among the enlisted. Perhaps the warrant officer metaphoric stereotype would suit the best as is a broadly appealing solution to such an 'in-between' status that has been adopted worldwide. While it may be tempting to relegate the Cowboys to being 'power users', the study has shown a greater degree of sophistication, impact, and cause for responsibility and accountability than this.

Notes

1. For the purpose of this paper, we use the term Low Code Development to denote Robot Process Automation (RPA) as well. Both LCD and RPA are high-level application development platforms, but they work in different ways and are used in different contexts. Space does not allow for a further elaboration on this point.

References

Alexander, F. (n.d.). What Is Low-Code? [2021 Update]. Outsystems.Com. https://www.outsystems.com/blog/posts/what-is-low-code/

Binzer, B., & Winkler, T. J. (2022, October). Democratizing Software Development: A Systematic Multivocal Literature Review and Research Agenda on Citizen Development. In *International Conference on Software Business* (pp. 244-259). Cham: Springer International Publishing.

Binzer, B. (2023). Low-Coders, No-Coders, and Citizen Developers in Demand: Examining Knowledge, Skills, and Abilities Through a Job Market Analysis.

Burrell, G., Morgan, G., Burrell, G., & Morgan, G. (1979). Assumptions about the nature of social science. Sociological paradigms and organisational analysis, 248(1), 1-9.

Bexiga, M., Garbatov, S., & Seco, J. C. (2020). Closing the gap between designers and developers in a low code ecosystem. Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, 5–15. https://doi.org/10.1145/3417990.3420195

Clark, L. (2019, November 18). Low-code maturity boosts efficiency and helps user acceptance. ComputerWeekly.Com. https://www.computerweekly.com/feature/Low-code-maturity-boosts-efficiency-and-helps-user-acceptance

Creswell, J. W. (1994). Mixed-method research: Introduction and application. In Handbook of educational policy (pp. 455-472). Academic press.

Elshan, E., Dickhaut, E., & Ebel, P. A. (2023). An investigation of why low code platforms provide answers and new challenges *56th Hawaii International Conference on System Sciences*.

Eulerich, M., Waddoups, N., Wagener, M., & Wood, D. A. (2024). The Dark Side of Robotic Process Automation (RPA): Understanding Risks and Challenges with RPA. *Accounting Horizons*, 1-10.

Friedman, A. L., & Cornford, D. S. (1989). Computer Systems Development: History Organization and Implementation. John Wiley & Sons, Inc.

Heijmans, Jeroen (2016) "Low Code: wave of the future or blast from the past?", Software Improvement Group.

Humphrey, W. S. (1989). The software engineering process: definition and scope. ACM SIGSOFT Software Engineering Notes, 14(4), 82–83. https://doi.org/10.1145/75111.75122

Kautz, K., Madsen, S., & Nørbjerg, J. (2007). Persistent problems and practices in information systems development. Information Systems Journal, 17(3), 217–239. https://doi.org/10.1111/j.1365-2575.2007.00222.x

Khorram, F., Mottu, J. M., & Sunyé, G. (2020). Challenges & opportunities in low-code testing. Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings. Published. https://doi.org/10.1145/3417990.3420204

Lethbridge, T. C. (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: 10th International Symposium on Leveraging Applications of Formal Methods, ISoLA 2021, Rhodes, Greece, October 17–29, 2021, Proceedings 10 (pp. 202-212). Springer International Publishing.

Martins, R., Caldeira, F., Sa, F., Abbasi, M., & Martins, P. (2020). An overview on how to develop a low-code application using OutSystems. 2020 International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE), 395–401. https://doi.org/10.1109/icstcee49637.2020.9277404

Naqvi, S. A. A., Zimmer, M. P., Syed, R., & Drews, P. (2023). Understanding the Socio-Technical Aspects of Low-Code Adoption for Software Development.

Microsoft. (n.d.). Business Application Platform | Microsoft Power Platform. Microsoft.Com. https://powerplatform.microsoft.com/en-us/

Noppen, P., Beerepoot, I., van de Weerd, I., Jonker, M., & Reijers, H. A. (2020). How to keep RPA maintainable?. In Business Process Management: 18th International Conference, BPM 2020, Seville, Spain, September 13–18, 2020, Proceedings 18 (pp. 453-470). Springer International Publishing.

Ngwenyama, O. K. (1993). Developing end-users' systems development competence: an exploratory study. Information & Management, 25(6), 291-302.

Oltrogge, M., Derr, E., Stransky, C., Acar, Y., Fahl, S., Rossow, C., Pellegrino, G., Bugiel, S., & Backes, M. (2018). The Rise of the Citizen Developer: Assessing the Security Impact of Online App Generators. 2018 *IEEE Symposium on Security and Privacy* (SP). Published. https://doi.org/10.1109/sp.2018.00005

Osmundsen, K., Iden, J., & Bygstad, B. (2019). Organizing robotic process automation: balancing loose and tight coupling.

Pinho, D., Aguiar, A., & Amaral, V. (2023). What about the usability in low-code platforms? A systematic literature review. Journal of Computer Languages, 74, 101185.

Ploder, C., Bernsteiner, R., Schlögl, S., & Gschliesser, C. (2019). The Future Use of LowCode/NoCode Platforms by Knowledge Workers – An Acceptance Study. Communications in Computer and Information Science, 445–454. https://doi.org/10.1007/978-3-030-21451-7_38

Rashid, F. Y. (2021, April 28). Gartner says low-code, RPA, and AI driving growth in 'hyperautomation.' VentureBeat. https://venturebeat.com/2021/04/28/gartner-says-low-code-rpa-and-ai-driving-growth-in-hyperautomation/

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). Published. https://doi.org/10.1109/seaa51224.2020.00036

Sadovnikov, K., Sweijen, R., van der Werf, J. M., & van de Weerd, I. (2023). A Framework to Assess the Suitability of Low-Code for BPM.

Virta, T. (2018). Relation of low-code development to standard software development: Case biit oy.

Yin, R. K. (2009). Case study research: Design and methods (Vol. 5). sage.