9-2024

Low/no-code and traditional code integration in digital banking

Kim Siang YEO
Singapore Management University, ks.yeo.2021@mitb.smu.edu.sg

Alan @ Ali MADJELISI MEGARGEL
Singapore Management University, ALANMEGARGEL@SMU.EDU.SG

Published in ***Journal of Digital Banking*** **Vol. 9, 2** 172–188 (2024) DOI: 10.69554/XLSZ2195

Low/no-code and traditional code integration in digital banking

Received (in revised form): 12th February, 2024

Yeo Kim Siang

Postgraduate Student, School of Computing and Information Systems, Singapore Management University, Singapore

Yeo Kim Siang is a postgraduate student, Master of IT in Business, at Singapore Management University. Relatively new to the working world, Kim Siang has three years of experience in software development, data engineering and data analytics, specifically in the banking and FinTech industries. Currently, he is balancing management roles as a project manager and technical roles as a data analyst and engineer, managing his company's latest migration efforts from its legacy systems into a new PySpark-supported system.

School of Computing and Information Systems, Singapore Management University, 80 Stamford Road, Singapore 178902
Mob: +65 97280011; E-mail: ks.yeo.2021@mitb.smu.edu.sg

Alan Megargel

Associate Professor of Information Systems (Practice), School of Computing and Information Systems, Singapore Management University, Singapore

Alan Megargel, PhD, is Associate Professor of Information Systems (Practice) at Singapore Management University, where he serves as Coordinator for the undergraduate Financial Technology Career Track. His current areas of specialisation include enterprise architecture in banking, service-oriented architecture (SOA), payments technology and nonbank FinTech alternative financial services. Alan has 30 years of industry experience, having served as Chief Technology Officer at TIBCO Software Asia, Vice President and Head of SOA at OCBC Bank and Senior Enterprise Architect at ANZ Bank. His banking technology experience covers retail and corporate banking, Basel II, data warehouse, data centre operations and technology infrastructure. Alan holds a Doctor of Innovation from Singapore Management University and a Master of Science in Software, Systems and Information Engineering from the University of Sheffield.

School of Computing and Information Systems, Singapore Management University, 80 Stamford Road, Singapore 178902
Tel: +65 6808 5276; Mob: +65 9669 0924; E-mail: alanmegargel@smu.edu.sg

Abstract This paper seeks to combine the merits of low/no-code programming (LNCP) with traditional programming (TP) systems for increased agility in digital banking software development. While it is easy to fall prey to shiny object syndrome in today's dynamic banking technology landscape, it is not easy to select the right technology to suit the current and future needs of the financial industry. Instead, LNCP makes it possible to lower the technical entry barriers to technology development. Integrating TP with LNCP, when needed, compensates for the shortcomings related to LNCP and provides digital banks with a more comprehensive software development approach. The adoption of this approach improves time-to-market of new innovative financial solutions. There has been little progress in this direction, academically or in practice. This paper includes an empirical study, interviews with banking professionals, on the merits of LNCP and TP, as well as an experimental project implementation that integrates LNCP and TP in the development of a retail Internet banking application. In the context of digital banking solution development, the result of the experiment reveals the merits of LNCP/TP hybrid systems in terms of agility, scalability, change management and cost-effectiveness.

KEYWORDS: low/no-code programming, LNCP, software development, agility, digital banking, Internet banking

DOI: 10.69554/XLSZ2195

INTRODUCTION

Banks have played a historically, and at times contentiously, critical role in the smooth running of local and global economies. A powerful institution, however, remains relatively powerless in the face of the very entity that empowered it — technology. Technology, when propelled by the deep pockets in the financial industry, progresses rapidly. The same technology that banks use to smoothen their processes and leave their competitors behind is holding them back and empowering their smaller and more agile start-up counterparts today.

Change, however, is good. Banks form the pillars that support local and global systems for reasons that far surpass their technology or surface functions. They do so because they inspire trust everywhere and anywhere. The connection with government, society and commercial institutions makes banks a keystone of our world today. Banking technology has long taken a trial-and-error approach, conducted by an array of large institutions to establish, something that the everyday start-up cannot guarantee it can replicate or achieve quickly enough to replace banks.

This paper posits that a new paradigm of software development that combines the best of low/no-code programming (LNCP) and traditional programming (TP), as a means to revamp banking technology, will help banks to improve their time-to-market deployment of new innovative digital banking solutions. First, we review the relevant academic literature before introducing some essential concepts and techniques. Secondly, we analyse the results of an empirical study involving interviewees from the banking industry. Finally, we report on the results of an experimental project to develop an LNCP/TP hybrid system, a retail Internet banking application.

RELATED WORK

Much of the previous work covers LNCP as a stand-alone system in the following ways. Some papers describe specific LNCP and their respective tools. In this context, researchers discuss the LNCP Aurea BPM,[1] RESTsec,[2] Sagitec Software Studio[3] and Smart Maker Authoring,[4] as well as the low-code development tools Xatkit[5] and vf-OS.[6] Other papers discuss programming languages for LNCP, testing processes and usability problems. Khorram *et al.* carried out an analysis of the testing components of five LNCPs, and, based on their analysis, they propose a feature list with possible values for low-code testing.[7] Ragusa and Henriques present the web-based tool, VPLreviewer, which enables the code review of visual programming languages (VPL), as well as focusing on the domain-specific language used in OutSystems to develop web and mobile applications.[8]

Application-based LNCP research papers, while fewer, remain quite comprehensive, especially for the banking sector. The application areas vary between manufacturing,[9,10] security[11] and (web)-application development.[12-14] Furthermore, Heffner and Mettrick summarise the capabilities and benefits of a potential LNCP for financial institutions.[15] For the banking sector and its customers, where security, compliance and quick adaptation to the respective needs are of great importance, an LNCP would fully cover the requirements.[16] Moreover, Sahay *et al.* conducted a technical study on eight LNCPs to enable potential customers to find the most suitable platform for their specific requirements.[17] Researchers state that handing over processes to business departments with the help of LNCP can reduce costs and implementation time and thus better

meet the respective needs.[18,19] An efficiency analysis shows that companies that use LNCP have greater freedom to adapt their activities to constantly changing market conditions.[20] Nevertheless, important future challenges of LNCP can be inoperability, extensibility, steep learning curves and scalability[21] but also the integration of machine learning (ML) and Internet of things (IoT),[22] and artificial intelligence (AI).[23]

Previous studies have been more hypothetical and experimental, where much of the practical uses of LNCP applications have been explored without external research and publication in mind, and the banking context is missing. While discussing the potential good that LNCP can bring to banking is beneficial, it is difficult to propose LNCP to management without any concrete results. This paper strives to do just that by reiterating the benefits of LNCP in banking while empirically replicating a representative banking system to demonstrate those benefits.

ESSENTIAL CONCEPTS AND TECHNIQUES

Banking and economy

Cross-country empirical evidence over the last few decades suggests that the development of the financial system, principally banks, stimulates economic growth, and this is known as the finance-growth nexus.[24-26] They act as safe havens for depositors and are significant sources of credit for households, small- and medium-sized firms, corporations and governments. Moreover, banks create liquidity for the non-bank public by transforming relatively illiquid assets such as loans to informationally opaque businesses into relatively liquid liabilities such as transaction deposits that allow almost instantaneous access to funds. Banks also create significant liquidity by issuing off-balance-sheet guarantees like loan commitments that allow customers to draw funds under predetermined conditions.[27,28] Banks also manage credit, solvency, interest rate, foreign exchange rate, liquidity and other risks via diversification, derivatives and other on- and off-balance sheet activities. Without banks and other financial services providers, entrepreneurs could only start new businesses that drive innovation and economic growth if they were born rich or accumulated capital over time.

Banking and technology

The first core banking system appeared in the 1970s.[29] The legacy core banking systems have a monolithic architecture, comprising tightly coupled components relying on shared resources such as a single code base, databases and servers. While these systems provide a robust and secure architecture for systems transactions, they hurt modularity, scalability and flexibility.[30] To meet the increasing and changing needs of the market, monolithic core banking systems have been modified excessively and deviated from the intended architecture over time.

Two popular paradigms taking over monolithic systems are SOA and, most recently, microservice architecture.[31] While both paradigms are rooted in the principle of modularity, various approaches have been proposed in the literature for changing the legacy.[32,33] Reviewing these approaches, however, indicates a certain degree of confusion regarding conceptual overlap and change outcomes. In particular, insufficient consideration has been given to highlighting organisational-level and strategic change-related aspects of legacy systems modernisation. For instance, with a few exceptions,[34] previous studies have overlooked modernisation of nontechnical barriers to legacy systems, primarily related to the banking industry.

Banking and FinTech

The Financial Stability Board (FSB) defines financial technology (FinTech) as 'technologically enabled financial innovation that could result in new business models, applications, processes, or products with an

associated material effect on financial markets and institutions, and the provision of financial services'.[35] FinTech providers are developing new services and products in the financial services sector to aggregate the functions of banks to their customers. If they succeed, this will surely change the existing business landscape of financial services. Traditional banking institutions can do the same with their significant capital resources. In contrast, FinTech providers cannot replicate the advantages of large banking institutions in terms of trust and reputation, regulatory compliance and range of services.[36]

Low/No-code programming

In 2014, Forrester Research introduced the term *low/no-code programming* (LNCP), which is described as software development with minimal source code using interactive graphical interfaces to simplify complexity.[37] LNCPs, which are often 'products and cloud services',[38] follow a product-as-a-service (PaaS) model.[39] They encourage visual development through declarative techniques for defining an application's user interface, business logic and data model.[40] LNCP significantly changes the way applications are developed. It shifts from a traditional IT-driven process involving manual coding to a more business-focused approach that uses visual drag-and-drop functions.[41] This makes it possible for even nonprofessional developers to create applications with minimal training time.

Traditional programming

TP involves tasks such as analysis, generating algorithms, profiling algorithms' accuracy and resource consumption and implementing algorithms (usually in a particular programming language, commonly called coding). Tasks accompanying and related to programming include testing, debugging, source code maintenance, implementation of build systems and management of derived artefacts, such as the machine code of computer programs.

INTERVIEWS WITH INDUSTRY PROFESSIONALS

There is no lack of theoretical research on LNCP and TP in the literature. It is difficult, however, to find material on how it can be applied to the financial industry by professionals in the finance industry. While an extensive empirical survey is beyond the scope of this paper, a preliminary study has been provided that involves interviews with banking industry professionals. Banking infrastructure is usually implemented by either an in-house team or an external vendor. To establish a comprehensive understanding of the full suite of software used, software developers employed by banks and employees of popular vendor solution providers were interviewed. Current or previous banking professionals with experience in other facets of the finance industry were also included on the expectation that the broader professional experience would provide insights that banking-only professionals might not have. Thus, three types of interviewees were selected, namely vendors, banking-only professionals and finance professionals. Table 1 lists the interview subjects.

The interview results are summarised in Tables 2 and 3, grouped by advantages and challenges of LNCP.

Despite the challenges, LNCP remains a compelling option for banks. The need for agility, cost-effectiveness and the involvement of nontechnical stakeholders in the development process is increasing. Banks, however, need to approach LNCP adoption strategically. LNCP can coexist with TP approaches, allowing banks to leverage the strengths of both methods. This hybrid approach allows banks to harness the advantages of LNCP while addressing their specific limitations, ensuring a balanced and pragmatic approach to technology adoption in the banking sector.

Table 1: Interview subjects.

Category	Occupation	Description
Vendor	Consultant	Both individuals work closely with banking clients and actively implement IT projects for them. While they might not give an overarching view of the types of systems a bank uses, they can give us a glimpse into the types of projects a bank outsources, for what sort of software and why.
	Presales data scientist	
Banking-only professionals	Data analyst	All individuals directly interface with the bank's IT infrastructure. Using their interviews, we can extrapolate the types of systems and languages used in most banks.
	Software developer (frontend)	
	Full stack developer	
Finance professionals	Information security analyst	The people interviewed have served extensively in the banking industry. They can provide a new perspective that relates to banking. Their experience can also be used to extrapolate the types of systems and languages used most in a bank.
	Cloud DevOps engineer	
	Cloud and DevOps lead	
	Risk infrastructure	
	Software engineer	
	AI/ML specialist	While not an obvious pick for a study related to banking, this individual's experience working closely with the company's finance department implementing AI solutions gives us a sneak peek into how technology in a bustling start-up is structured. This can inform us of gaps in a bank's technology infrastructure and provide ways to improve the newly suggested paradigm.

Table 2: LNCP advantages.

LNCP advantage	Description
Efficiency and speed	LNCPs enable rapid application development, allowing banks to respond quickly to changing market demands and regulatory requirements. This agility can be a competitive advantage in the fast-paced banking industry.
Customisation and control	LNCPs offer a balance between customisation and control. Banks can tailor solutions to their needs while maintaining governance and compliance standards.
User-friendly interface	The user-friendly nature of LNCPs empowers nontechnical professionals within the bank to actively participate in software development. Bridging the gap between business and technology teams can improve collaboration and understanding.
Integration capabilities	LNCPs often come with pre-built connectors and application programming interface (API) s, simplifying the integration of banking systems with other applications and services, and enhancing interoperability.
Overcoming resistance to change	LNCPs can help banks overcome resistance to technological change by involving a broader spectrum of employees in the development process, including those who may not have traditional coding skills.

Table 3: LNCP challenges.

LNCP challenge	Description
Compliance and security	Banking operations are highly regulated, and LNCPs must meet stringent compliance and security requirements. Banks need to assess whether LNCPs can adhere to these standards carefully.
Complexity and scalability	While LNCPs excel in rapid development, they may face challenges in handling complex, large-scale banking systems. Banks with intricate operations may find LNCPs unsuitable for specific mission-critical applications.
Vendor lock-in	Banks should be cautious of potential vendor lock-in when adopting LNCPs. The choice of an LNCP provider should consider long-term strategies and the ability to migrate to other systems if necessary.

LOW/NO-CODE PROGRAMMING VERSUS TP

Comparison

LNCP represents a paradigm shift in software development, offering a user-friendly and visually intuitive approach to creating applications.[42] In contrast to traditional software development, which relies heavily on manual coding, LNCP empowers citizen developers to build applications using drag-and-drop modellers and pre-built components. While both LNCP and traditional development have their merits, understanding the important differences in various aspects is crucial for organisations and developers when choosing the right approach for their needs. These differences are closely examined in what follows to get a clearer view of how LNCP compares with TP across a range of crucial criteria.

Development approach

LNCP relies on a visual and user-friendly approach, allowing citizen developers to create applications using drag-and-drop modellers and intuitive tools. In contrast, it relies on manual coding, utilising web frameworks and programming languages, often requiring advanced coding skills. This fundamental difference in approach makes LNCP accessible to a broader range of users.

Speed of development

LNCP excels in rapid application development, enabling the creation of functional applications in less than two months. This agility is highly beneficial for quick experimentation and deployment, making LNCP a popular choice for start-ups. In contrast, TP typically takes a minimum of six months to one year to fully develop an application, depending on project complexity, which can result in a longer time-to-market.

Customisation

While LNCP tools offer a user-friendly environment for application creation, they may come with limitations in terms of customisation. Some LNCP solutions, however, provide the flexibility to incorporate custom user interface components. In contrast, TP offers extensive customisation capabilities, allowing applications to be finely tailored to specific requirements.

Agility

LNCP platforms offer exceptional agility, allowing quick changes, rapid app releases, easy feature additions and efficient error corrections. This agility is a significant advantage, particularly in dynamic business environments. In TP, agility depends more on the team's size and processes, potentially resulting in time-consuming modifications and project scope adjustments.

Deployment

LNCP expedites deployment, as applications can be created with minimal coding, and many platforms include built-in DevOps and hosting infrastructure. This simplifies the deployment process and reduces the time to make applications accessible to users. In contrast, traditional development requires building applications from the ground up, which can lead to longer deployment times, especially if DevOps and hosting infrastructure must be assembled separately.

Quality

LNCP platforms maintain application quality through extensive integration, standard performance and live-debugging options. This ensures that applications remain error-free and perform optimally. Traditional development offers scalability and outstanding performance but may require

more time for rigorous testing and debugging to achieve the same level of quality assurance.

Maintenance

Maintenance is simplified in LNCP, as the platform often handles crucial tasks such as security, maintenance and upgrades. This reduces the burden on development teams and ensures that applications remain up-to-date and secure. In traditional development, a dedicated team is typically required to manage regular updates and maintenance tasks, which can be resource intensive.

Template availability

LNCP offers many pre-built templates and components, expediting the application development process. These templates serve as building blocks, making it easier for users to create applications with specific functionalities. In traditional development, applications are built from scratch without the availability of pre-made templates, which can extend the development timeline.

Scalability

LNCP platforms excel in scalability, accommodating variable workloads and organisational growth with ease. They are designed to handle increased user demands and can scale applications without significant development effort. In traditional development, achieving scalability can be challenging without the expertise of experienced software developers, potentially leading to performance issues during periods of growth.

Security

LNCP solutions often comply with ISO 2007 and SOC2, ensuring data security and adherence to essential data protection regulations. This built-in security is a significant advantage. In traditional development, security measures must be implemented by the software development team, which can take time and potentially compromise the software's quality during implementation.

Multiplatform capability

LNCP applications are versatile and capable of running on mobile, web and cloud platforms, providing a seamless user experience across various devices. This multiplatform compatibility is a valuable feature. In traditional development, developers often choose between native or cross-platform development methods to ensure their applications work on multiple platforms, which can involve additional development effort.

Impact of generative AI

Generative AI (Gen-AI) tools can assist in TP by generating code fragments, or even entire code modules, given a set of well-defined requirements expressed as engineered prompts. Gen-AI can reduce the need for manual coding, allowing developers to focus more on higher-level creativity and problem-solving, ultimately enhancing the software development process.

By definition, LNCP achieves some of the same objectives as Gen-AI in this context, in that manual coding is reduced or eliminated altogether. Leading LNCP such as OutSystems feature AI-enabled 'mentors', which act as 'experts throughout the software development lifecycle, guiding, automating, and validating the work of developers', increasing the productivity of developers (https://www.outsystems.com/evaluation-guide/ai/).

Current issues with LNCP

It seems LNCP can do almost everything TP can and, at times, more. It is largely unknown, however, and has not been widely adopted in any industry, including banking. The principal reasons why some organisations

do not adopt low-code development include a lack of knowledge of LNCP (47 per cent), apprehensions about vendor lock-in with LNCP (37 per cent), apprehensions about the scalability of LNCP (28 per cent) and apprehensions about the security of LNCP (25 per cent).[43]

Limited customisability/flexibility

The visualised building blocks in low-code platforms are pre-implemented and fixed in most cases.[44] Such inflexibility makes the applications less customisable than those developed by traditional coding development.[45] It will be difficult and time-consuming to develop complicated or customised features or functionalities not provided on the low-code platforms.[46] Implementing these desired features using codes and integrating them into low/no-code applications is an approach that lacks consistency and efficiency.[47] Low/no-code platforms usually outperform the traditional development process in implementing simple applications where the predefined components address everyday needs or processes well.[48] When it comes to projects such as highly customised applications, data science models or data science workflows, however, low/no-code platforms are not customisable enough for these tasks.[49]

Limited scalability

Most current low/no-code platforms are mainly used to develop small-scale applications. In contrast, they are seldom used for large-scale, complex or crucial business applications owing to their limited scalability.[50] According to Rymer and Richardson, the average runtime scale of applications reported by low-code platform providers was between 200 and 2,000 concurrent users.[51]

Security concerns

Since most low/no-code platform users hardly do or cannot customise the applications, they must completely trust that the services do not generate vulnerabilities that cause bugs or data leaks.[52] For example, Mobincube, a paid low-code service, tracked users silently through Bluetooth low-energy beacon without clearly declaring this in the terms and conditions.[53] Suppose organisations are dependent on their low/no-code platform vendors. In that case, their data might be vulnerable to data breaches since organisations do not fully control data security and source code.[54,55] Moreover, if the platform vendors wind up, there will not be further security updates, and organisations cannot fix new security flaws later.[56]

Vendor lock-in

LNCP platforms come with a degree of vendor lock-in as users commit to a specific platform for application development. The extent of this lock-in depends on the level of integration and reliance on the chosen LNCP. In contrast, traditional development experiences minimal lock-in, primarily because it often involves using open-source programming languages and software, providing greater flexibility and independence.

TECHNOLOGY IN BANKING

Most banks divide themselves into departments and pillars based on clientele, namely institutional and consumer banking; they unite in each entity's functions. Since banking operations are complicated, extensive and often opaque, it would help to use a relatively more straightforward aspect of a bank to run our experiments since the lessons learned there can be applied to other parts of the bank too.

Retail banking has been chosen as our context. Operations in retail banking tend to be more straightforward, given how the clientele are consumers with less funding and professional requirements compared with larger institutions. Specifically, we are looking at digital banking, rather than

Table 4: Digital banking functions and software development tools.

Credit, deposit and capital-raising services	Payments, clearing and settlement services		Investment management services
	Retail	**Wholesale**	
Lending marketplaces	Mobile wallets	Value transfer networks	Copy trading
Mobile banking	Peer-to-peer transfers	FX wholesale	E-trading
Credit-scoring	Digital currencies	Digital exchange platforms	Robo-advice
Crowdfunding			High-frequency trading
Software development tools			
LNCP	Frontend and backend: OutSystems, Mendix, Microsoft Power Apps, Bubble.io		
TP	Frontend: HTML, CSS, JavaScript Backend: Java (Spring Boot), Python		

brick-and-mortar banking, given the new age of digitalisation. More importantly, a study of a new software paradigm would not be possible without software. Hence, retail Internet banking software has been chosen for our experimental test bed. A high-level view of digital banking functions, and software development tools used, is shown in Table 4.

Digital banking functions

There are four large factions in the types of banking functions. *Credit, Deposit and Capital-Raising Services* provide loans, accept deposits and help businesses raise capital through stock offerings and bonds. *Payments, Clearing and Settlement Services* enable transactions, clear payments and ensure the secure settlement of financial agreements. This can be done on a retail level, on a customer-by-customer basis, at bulk or on a wholesale level. *Investment Management Services* assist individuals and institutions in managing their investments and offer digital platforms for portfolio management.

Software development tools

A large part of any bank's infrastructure remains with TP. Given the massive amount of code, most software developers inherit instead of creating code. This means that banks are forced to maintain legacy code written using TP. Many core banking functions tend to be written in pre-Internet-era TP, but some banks have moved to modern languages such as C#, Java and Python. These languages provide additional features, such as integrations with a more extensive array of software or a more stable compiling system with the same stability needed for enterprise code. Internet-era languages, such as HTML, CSS and JavaScript, are used for web design interfaces. The main difference lies in the type of framework used and the design principles implemented.

Emergence of LNCP as an alternative to TP

Alluding to the reasons for the lack of LNCP use, IT departments today tend to be TP-heavy. IT places more weight on the benefits of customisability, scalability and security and less on the costs of a slow deployment period. That cost-benefit scale, however, might tip against TP and towards LNCP with the incoming cloud-heavy world.

Cloud-native application development is one of the fastest-growing trends. According to Gartner, 95 per cent of applications will become cloud-native by 2025, and more than 85 per cent of organisations will need to use cloud-native technologies to execute their digital strategies fully.[57] The decisive

push towards cloud implementation is undeniable. Cloud-native technologies also come with built-in scale and security, traits that are shared with TP. Ease of use of cloud infrastructure will make organisations, including banks, more competitive. The new cloud-driven IT world aids in, and will eventually mandate, more reactive applications and faster deployment times, traits that TP lacks and LNCP excels in.[58]

A complete conversion from TP to LNCP would involve a total overhaul on many levels — technical, personal and infrastructural — rendering obsolete the traditional skills built by institutions and people over time. Instead, we should look for ways in which the adoption of LNCP will exponentially enhance an institution's offering. Most LNCPs allow additional TP integrations, allowing new and current users to have the best of both worlds. This allows institutions, including banks, to pull forward by leveraging on LNCP in a manner that allows them to retain their previous TP advantages.

EXPERIMENT: LNCP AND TP INTEGRATION IN A RETAIL INTERNET BANKING APPLICATION

This section documents our experience replicating a relatively more manageable Singapore Management University (SMU) banking system, SMU Teaching Bank (a.k.a. SMU tBank), comprising all retail Internet banking core functions implemented within an integrated LNCP/TP system.

First, we identified a domain-relevant case already built, the SMU tBank Retail Internet Banking (RIB) application, developed by SMU faculty and students. RIB has seven main features, each with its corresponding sub-sections: *Home, View Accounts, Fund Transfer, Loan, Wealth Management, Book Appointment and Profile/Logout*. Figure 1 shows the SMU tBank RIB loan repayment page, which serves as the landing page for any customer after login.

Secondly, to be named the SMU tBank OutSystems Experiment, this experiment aimed to determine whether an LNCP/TP

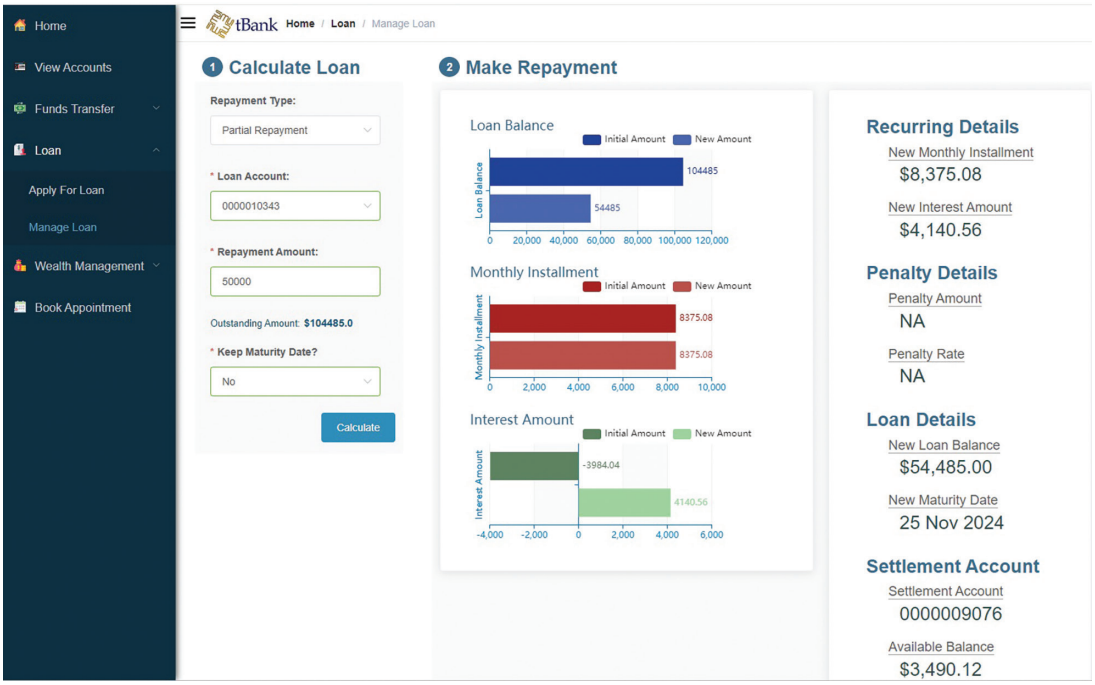

Figure 1 SMU tBank RIB loan repayment page

hybrid approach would provide the same flexibility, stability and customisability as a TP approach while reducing development time. A regular software development cycle was followed, changing only how software is written. In the original software development cycle, Java was used as the backend language to build the logical processes, JavaScript was used to build the frontend and GitHub was used for versioning control. In our experimental build, versioning and coding were entirely covered by the OutSystems LNCP.

Lastly, we compared the labour, maintenance effort, time and skill set required between the original TP implementation and the experimental LNCP implementation. We also determined the degree of visuals and functions replicated by the LNCP implementation to determine the effectiveness of the new LNCP/TP hybrid system.

LNCP candidates

There are many platforms for LNCP-driven software development, the market leaders being OutSystems, Mendix, Microsoft Power Apps and Bubble.io. We compared them to find the most appropriate LNCP for an enterprise application.

OutSystems

OutSystems allows you to develop and deploy applications quickly and efficiently. It provides a visual development environment, comprehensive features and capabilities and a large community of users and developers. OutSystems is used by many organisations, including Fortune 500 companies, government agencies and start-ups.

Mendix

Mendix allows you to develop and deploy applications quickly and efficiently. It is similar to OutSystems regarding features and capabilities but is generally more straightforward. Mendix is also used by many organisations, including Fortune 500 companies, government agencies and start-ups.

Microsoft Power Apps

Microsoft Power Apps is a part of the Microsoft Power Platform. It allows you to develop and deploy simple to medium-complexity applications quickly and efficiently. Microsoft Power Apps is a good choice for businesses and organisations already using Microsoft products and services.

Bubble.io

Bubble.io allows you to develop and deploy applications of simple to medium complexity quickly and efficiently. It is the most affordable low-code platform and also the easiest to use. Bubble.io is a good choice for businesses and organisations that are on a tight budget or that do not have much technical expertise.

LNCP candidate feature comparison

Table 5 provides a comparison of LNCP candidate features.

Table 5: Comparison of LNCP candidate features.

Feature	OutSystems	Mendix	Microsoft Power Apps	Bubble.io
Performance and scalability	High	Medium	Medium	Low
Flexibility and extensibility	High	High	Medium	Low
Ease of use	Easy	Medium	Easy	Easy
Large community and ecosystem	Yes	Yes	Yes	Yes
Pricing	Expensive	Medium	Medium	Affordable

Performance and scalability

OutSystems and Mendix use model-driven software engineering to generate native code for the target platform. This means that OutSystems and Mendix applications are typically faster and more scalable than applications developed using other low-code platforms.

Flexibility and extensibility

OutSystems and Mendix provide several features that make them flexible and extensible platforms. For example, OutSystems and Mendix allow you to create custom components, extend the platform using Java and C# and integrate with third-party systems.

Ease of use

OutSystems, Mendix and Microsoft Power Apps provide visual development environments that make creating and managing applications easy. Bubble.io, however, is the most accessible platform because it does not require any coding knowledge.

Large community and ecosystem

OutSystems, Mendix and Microsoft Power Apps all have large and active communities of users and developers. There are also several third-party tools and integrations available for these platforms. Bubble.io also has a growing community, but it is not as large as those for the other platforms.

Pricing

OutSystems is the most expensive platform because it offers the most features and capabilities. Mendix and Microsoft Power Apps are less expensive but offer fewer features and capabilities. Bubble.io is the most affordable platform because it is newer with smaller features.

LNCP candidate selection

OutSystems was the ideal choice for our experiment; it excels in performance, scalability, flexibility and extensibility. According to our interviews with industrial professionals, OutSystems is highly prized within the software development community. While it is the most expensive, the system can be adopted by a large bank for an extended period, making it a prime candidate for benefitting from economies of scale and bulk pricing. OutSystems is the only LNCP evaluated here that can provide access to its core code through TP, precisely what we are testing for and promoting in this experiment. Most importantly, OutSystems is the leader in the low-code market. It has been around for over 20 years and has a proven track record of success with some leading financial institutions, including KeyBank, BBVA, Western Union, Santander Bank and CorporateOne.

SMU tBank OutSystems experiment

LNCP and TP are implemented as per the infrastructural design shown in Figure 2. The existing JavaScript user interface and business logic is replaced by an OutSystems implementation. The backend TP system remains and is supported by existing Representational State Transfer (REST) APIs called from the frontend user interface.

An example of a business logic implementation using OutSystems is shown in Figure 3, which implements the opening of a Dual Currency Deposit (DCD) account, a product offered under Wealth Management.

OutSystems allowed us to quickly build the business logic for each process by representing important entities or actions in the system with visual blocks and using arrows to design the business logic flow between blocks. Each flow can then be saved into a representative program. Each program can then be used across different web pages and even assimilated into other programs,

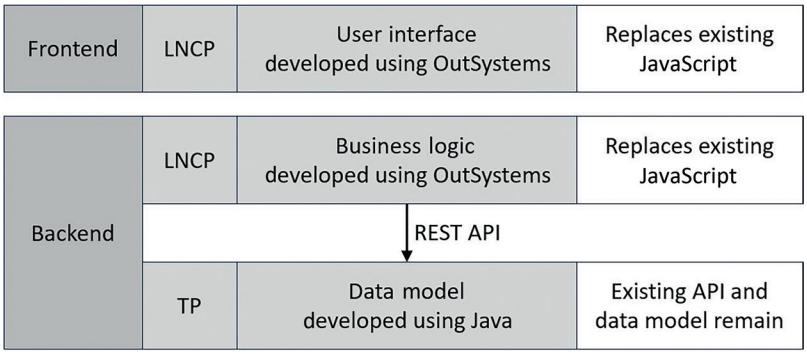

Figure 2 Infrastructural design of SMU tBank OutSystems experiment

Key: API, application programming interface; LNCP, low/no-code programming; TP, traditional programming.

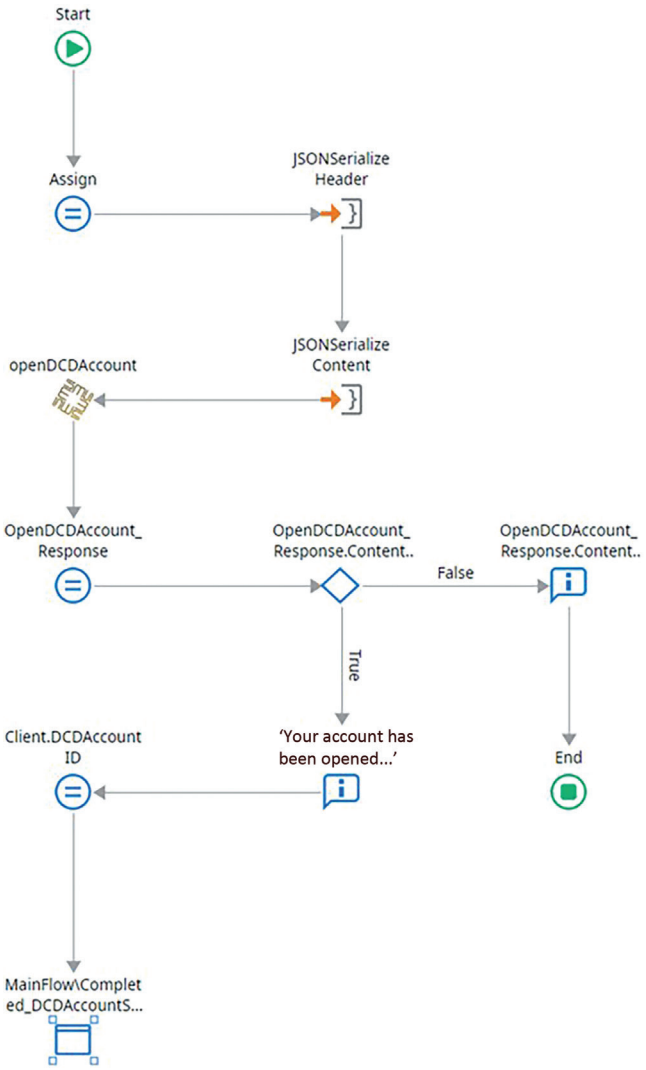

Figure 3 Example business logic implemented in OutSystems

dispensing with the need to create the same program repeatedly whenever we utilise it.

OutSystems allowed us to build web pages using the same drag-and-drop mechanism. To enforce abstraction, preset web design components were used for constructing web pages. We can choose web features available out-of-the-box from OutSystems, or we can customise our own. While the web features might seem limited, we can continually expand from the out-of-the-box features while complying with the organisation's web design standards. These new features can then be made available and reused by other software development projects.

Results and observations

Table 6 summarises our observations after replicating the SMU tBank RIB user interface and business logic using OutSystems, replacing the original TP implementation.

Implementation labour required

The original SMU tBank RIB development team comprised six students having four years of computer science education and was completed as a part of their final year project at SMU. Each student was allocated a particular role. The roles available were frontend software developer, backend software developer and project manager. In contrast, the SMU tBank OutSystems Experiment was run by just one person, having only two years of backend software development experience with no frontend software development or project management training.

Maintenance effort required

The original SMU tBank RIB project was written using open-source JavaScript libraries, which are susceptible to versioning changes done by external communities, changes that the team had little control over. In addition, any changes proposed required the correct individual with the appropriate skill set. Changes or bugs that happened to the user interface required a frontend software developer, while modifications to the business logic required the help of the backend software developer. In contrast, OutSystems provides a unified designer studio that makes it possible to edit web pages as well as the underlying business logic. One who understands OutSystems can make any changes, cutting down the required labour by at least half.

Time required

The original SMU tBank RIB project team took 10 months to complete the development of the user interface and business logic. In contrast, for the SMU tBank OutSystems Experiment, the user interface and business logic were replicated in exactly two months, for a time saving of eight months (ie it took one-fifth of the time).

Skill set required

Experience and expertise in JavaScript, HTML, CSS and web hosting were needed

Table 6: Observed performance of LNCP versus TP.

Feature	TP (Original implementation)	OutSystems (This experiment)
Implementation labour required	High	Low
Maintenance effort required	Medium	Low
Time required	High	Low
Skillset required	High	Low
Degree of visuals and functions replicated	Complete	80%–90% of original replicated

to develop the original SMU tBank RIB. In contrast, only basic knowledge of OutSystems was sufficient to replicate the SMU tBank RIB application in our experiment.

Degree of visuals replicated

Beyond minor colour scheming and layout issues, most of the important visuals and functions were replicated. The only difference between the original SMU tBank RIB and the SMU tBank OutSystems Experiment would be the stock historical price chart. The API to get any stock's information returned a string that looked like a JSON time series. Using JavaScript in this case, TP would have easily cleaned and converted it into an actual JSON time series object that could be parsed and used to replicate the visual. OutSystems did not have such a built-in function, making replicating this visual more complicated.

CONCLUSION

In conclusion, the research presented in this paper underscores the transformative potential of LNCP/TP hybrids as a new paradigm for software development within the banking industry. By carefully examining the capabilities, advantages and empirical evidence, we have demonstrated that these systems represent a compelling solution for addressing the dynamic and evolving needs of the sector. The important findings of this study emphasise several crucial points as follows:

Agility

LNCP/TP hybrid systems enable rapid development and customisation of software solutions, facilitating a more agile response to changing market demands. Banking institutions can leverage these platforms to deploy new features, products and services at an accelerated pace, enhancing their competitiveness.

Scalability

The scalability of LNCP/TP hybrid systems ensures that banking organisations can grow their operations efficiently without the traditional constraints associated with code-intensive development. This scalability is essential in an industry where expansion and adaptation are constant imperatives.

Change management

The flexibility inherent in LNCP/TP hybrid systems enables banks to adapt swiftly to regulatory changes, ensuring compliance while minimising disruption to operations. This adaptability is particularly crucial in an industry subject to evolving compliance requirements.

Cost-effectiveness

By streamlining the development process and reducing the need for extensive coding and software developers, LNCP/TP hybrid systems can save costs in the long run, making them a cost-effective solution for banks of all sizes.

Final thoughts

It is evident that LNCP/TP hybrid systems promise to revolutionise how software is developed and maintained in the banking sector. These systems empower financial institutions to meet customer demands, navigate industry challenges and maintain a competitive edge in an ever-evolving landscape.

It is essential to acknowledge, however, that the successful implementation of LNCP/TP hybrid solutions requires careful planning, effective governance and ongoing collaboration among various stakeholders. Furthermore, the technology landscape continues to evolve, and staying current with advancements in low/no-code platforms and their integration with TP is essential for long-term success.

This research underscores the compelling case for adopting LNCP/TP hybrid systems in the banking industry. By embracing this new paradigm, financial institutions can position themselves as leaders in innovation, ensuring they are well equipped to meet the demands of the modern banking landscape while maintaining code security and cleanliness. This approach accelerates development and enables banks to remain agile, adaptable and customer-centric in an era of rapid technological change.

References

(1) Waszkowski, R. (2019) 'A Low-Code Platform for Automating Line Processes in Manufacturing', *IFAC-PapersOnLine*, Vol. 52, No. 10, pp. 376–81.
(2) Zolotas, C., Chatzidimitriou, K. C. and Symeonidis, A. L. (2018) 'RESTsec: A Low-Code Platform for Generating Secure By-Design Enterprise Services', *Enterprise Information Systems*, Vol. 12, No. 8-9, pp. 1007–33.
(3) Arora, R., Ghosh, N. and Mondal, T. (2020) 'Sagitec Software Studio (S3) — A Low Code Application Development Platform', in '2020 International Conference on Industry 4.0 Technology (I4Tech)', IEEE, Pune, India, pp. 13–17.
(4) Chang, Y.-H. and Ko, C.-B. (2017) 'A Study on the Design of Low-Code and No-Code Platforms for Mobile Application Development', *International Journal of Advanced Smart Convergence*, Vol. 6, No. 4, pp. 50–5.
(5) Daniel, G., Cabot, J., Deruelle, L. and Derras, M. (2020) 'Xatkit: A Multimodal Low-Code Chatbot Development Framework', *IEEE Access*, Vol. 8, pp. 15332–46.
(6) Sanchis, R., García-Perale, Ó., Fraile, F. and Poler, R. (2019) 'Low-Code as Enabler of Digital Transformation in Manufacturing Industry', *Applied Sciences*, Vol. 10, No. 1, p. 12.
(7) Khorram, F., Mottu, J.-M. and Sunyé, G. (October 2020) 'Challenges & Opportunities in Low-Code Testing', in 'Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings', IEEE, Pune, India, pp. 1–10.
(8) Ragusa, G. and Henriques, H. (October 2018) 'Code Review Tool for Visual Programming Languages', in '2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)', IEEE, Pune, India, pp. 287–8.
(9) Heffner and Mettrick, ref 1 above.
(10) Sanchis, García-Perale, Fraile and Poler, ref 6 above.
(11) Zolotas, Chatzidimitriou and Symeonidis, ref 2 above.
(12) Arora, Ghosh and Mondal, ref 3 above.
(13) Chang and Ko, ref 4 above.
(14) Daniel, Cabot, Deruelle and Derras, ref 5 above.
(15) Heffner, M. and Mettrick, G. (2020) 'Innovation, Insight, and Trust: Customer Experience Excellence Delivered Responsibly in a Digital World', *Journal of Digital Banking*, Vol. 4, No. 4, pp. 351–63.
(16) *Ibid*.
(17) Sahay, A., Indamatti, A., Di Ruscio, D. and Pierantonio, A. (August 2020) 'Supporting the Understanding and Comparison of Low-Code Development Platforms', in '2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)', IEEE, Pune, India, pp. 171–8.
(18) Waszkowski, ref 1 above.
(19) Sahay, Indamatti, Di Ruscio and Pierantonio, ref 17 above.
(20) Waszkowski, R. and Nowicki, T. (2020) 'Efficiency Investigation and Optimization of Contract Management Business Processes in a Workwear Rental and Laundry Service Company', *Procedia Manufacturing*, Vol. 44, pp. 551–8.
(21) Sahay, Indamatti, Di Ruscio and Pierantonio, ref 17 above.
(22) Arora, Ghosh and Mondal, ref 3 above.
(23) Chang and Ko, ref 4 above.
(24) Levine, R. (2005) 'Finance and Growth: Theory and Evidence', in Aghion, P. and Durlauf, S. (eds), 'Handbook of Economic Growth', Volume 1, Elsevier, North Holland, pp. 865–934.
(25) Demirgüç-Kunt, A. and Maksimovic, V. (2002) 'Funding Growth in Bank-based and Market-based Financial Systems: Evidence from Firm-Level Data', *Journal of Financial Economics*, Vol. 65, No. 3, pp. 337–63.
(26) Berger, A. N., Hasan, I. and Klapper, L. F. (2004) 'Further Evidence on the Link between Finance and Growth: An International Analysis of Community Banking and Economic Performance', *Journal of Financial Services Research*, Vol. 25, No. 2-3, pp. 169–202.
(27) Berger, A. N., and Bouwman, C. H. S. (2016) 'Bank Liquidity Creation and Financial Crises', Academic Press.
(28) Bouwman, C. H. S. (October 7, 2018) 'Creation and Regulation of Bank Liquidity', available at https://ssrn.com/abstract=3266406 (accessed 9th April, 2024).
(29) Hariharan, N. and Reeshma, K. (2015) 'Challenges of Core Banking Systems', *Mediterranean Journal of Social Sciences*, Vol. 6, No. 5, p. 24.
(30) *Ibid*.
(31) Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., *et al*. (2017) 'Microservices: Yesterday, Today, and Tomorrow', in Mazzara, M. and Meyer, B. (eds), 'Present and Ulterior Software Engineering', Springer, Cham, pp. 195–216.
(32) Baghdadi, Y. and Al-Bulushi, W. (2015) 'A Guidance Process to Modernize Legacy Applications for SOA', *Service Oriented Computing and Applications*, Vol. 9, No. 1, pp. 41–58.
(33) Gholami, M. F., Daneshgar, F., Beydoun, G. and Rabhi, F. (2017) 'Challenges in Migrating Legacy Software Systems to the Cloud — An Empirical Study', *Information Systems*, Vol. 67, pp. 100–113.

(34) Khadka, R., Batlajery, B.V., Saeidi, A. M., Jansen, S. and Hage, J. (2014) 'How Do Professionals Perceive Legacy Systems and Software Modernization?', in Proceedings of the 36th International Conference on Software Engineering, ACM, Hyderabad, India, pp. 36–47.
(35) Financial Stability Board (2023) 'FinTech', Financial Stability Board, available at https://www.fsb.org/work-of-the-fsb/financial-innovation-and-structural-change/fintech/ (accessed 9th April, 2024).
(36) Feyen, E., Frost, J., Gambacorta, L., Natarajan, H., and Saal, M. (2021) 'Fintech and the Digital Transformation of Financial Services: Implications for Market Structure and Public Policy', *Bank for International Settlements (BIS) Papers*, No. 117.
(37) Sanchis, García-Perale, Fraile and Poler, ref 6 above.
(38) Al Alamin, M. A., Malakar, S., Uddin, G., Afroz, S., Haider, T. B. and Iqbal, A. (May 2021) 'An Empirical Study of Developer Discussions on Low-Code Software Development Challenges', in '2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)', IEEE, Pune, India, pp. 46–57.
(39) Rymer, J. and Appian, K. (2017) 'The Forrester Wave™: Low-Code Development Platforms for AD&D Pros, q4 2017', Forrester Research, Cambridge, MA.
(40) Prinz, N., Rentrop, C. and Huber, M. (2021). 'Low-Code Development Platforms: A Literature Review', in 'AMCIS 2021 Proceedings', AMCIS, Pune, India.
(41) Beranic, T., Rek, P. and Hericˇko, M. (2020) 'Adoption and Usability of Low-Code/No-Code Development Tools', in 'Central European Con ference on Information and Intelligent Systems', Faculty of Organization and Informatics, Varazdin, pp. 97-103.
(42) Russo, D., Visaggio, F. and Lanza, M. (2023) 'Low-Code/No-Code Platforms: A Paradigm Shift in Software Development', IEEE Software.
(43) OutSystems (2019) 'The State of Application Development: Is IT Ready for Disruption?', available at https://www.outsystems.com/-/media/E0A6E7121AAD4A4C975828265B3639ED.ashx?mkt_tok=eyJpIjoiT1RsbU56azNNakJsWVRaaiIsInQiOiIyNlBCdGlrRnVHclVEY2c3TWtSSEUwNWtTU3FBVVE0M2gwK0xoSW0xaktSZ3dWS2t6amQxOFU2WlFCRllwR256aUhMTHVWa0ROSnZrU2tRUlZ4cTV5RFJXb2o5Wlphc21jaFR-4bXY4ZmU3U3BrTkFNMm1BZm9MWkNsRHg0YjlzayJ9 (accessed 22nd March, 2024).
(44) Woo, M. (2020) 'The Rise of No/Low Code Software Development — No Experience Needed?', *Engineering*, Vol. 6, No. 9, pp. 960–1, https://doi.org/10.1016/j.eng.2020.07.007.
(45) Tay, N. (5th April, 2021) '7 Pros and Cons of Low-Code/No-Code', Major Online Business and Marketing, available at https://blog.hslu.ch/majorobm/2021/04/05/7-pros-and-cons-of-low-codeno-code-ntsy-2-ua-192667621-1/ (accessed 11th December, 2021).
(46) *Ibid*.
(47) Brocoders Company (6th March, 2021) 'The Pros and Cons of Low-Code Development', Hacker Noon, available at https://hackernoon.com/the-pros-and-cons-of-low-code-development-4y2p33g9 (accessed 11th December, 2021).
(48) Sarabyn, K. (2021) 'What is Wrong with Low and No Code Platforms?', Pandium, available at https://www.pandium.com/blogs/whats-wrong-with-low-and-no-codeplatforms (accessed 11th December, 2021).
(49) *Ibid*.
(50) Sanchis, García-Perale, Fraile and Poler, ref 6 above.
(51) Rymer, J. R. and Richardson, C. (August 2015) 'Low-Code Platforms Deliver Customer-Facing Apps Fast, But Will They Scale Up?', Forrester, available at https://www.forrester.com/report/LowCode-Platforms-Deliver-CustomerFacing-Apps-Fast-But-Will-They-Scale-Up/RES122546 (accessed 22nd March, 2024).
(52) Oltrogge, M., Derr, E., Stransky, C., Acar, Y., Fahl, S., Rossow, C., *et al.* (2018) 'The Rise of the Citizen Developer: Assessing the Security Impact of Online App Generators', in '2018 IEEE Symposium on Security and Privacy (SP)', pp. 634–47, https://doi.org/10.1109/sp.2018.00005.
(53) *Ibid*.
(54) Tay, ref 45 above.
(55) Oltrogge, Derr, Stransky, Acar, Fahl, Rossow, *et al.*, ref 52 above.
(56) *Ibid*.
(57) Gartner (2021) 'Gartner Says Cloud Will Be the Centerpiece of New Digital Experiences', available at https://www.gartner.com/en/newsroom/press-releases/2021-11-10-gartner-says-cloud-will-be-the-centerpiece-of-new-digital-experiences (accessed 22nd March, 2024).
(58) OutSystems (13th January, 2023) 'How Low-code Helps Banks Stay Competitive', https://www.outsystems.com/blog/posts/low-code-banks-stay-competitive/ (accessed 22nd March, 2024).