Infosys cobalt

VIEW POINT

LCAP TO THE ENTERPRISE

NAVIGATING THE CHASM – PRACTITIONERS' VIEW

Abstract

Pressure on IT departments to accelerate digital transformation is pushing Low code application platforms (LCAP) into mainstream.

This rapid expansion has created a set of challenges including aggressive suppositions on platform maturity, gaps in skillset, smaller & faster version releases amongst others, forcing organizations to re-review technology roadmap.

An outlier this time is the bewildering range of options for a relatively inceptive technology, suggesting undertones of a strong industry led positive assessment & impending consolidation.

This paper focuses on gaps faced by low code no code developer of a large & distributed enterprise development team. Amongst the plethora of competing products, this paper uses Microsoft Power Platform for demonstrating status of these yearnings.

Propositions in this paper are based on work experiences of the author in executing a range of enterprise projects in traditional & low code engagements, and extrapolations of pre-LCAP concepts.

Infosys®
Navigate your next

Table of Contents

1. LCAP - Overview

Evolutionary timeline of software industry is replete with attempts to harness benefits of low code development. Full-scale traditional languages had low code aspects, e.g Java IDEs built Swing UI through drag & drop long back. Similarly content management systems like WordPress enabled building of websites & blogs without any coding. From the Microsoft camp, Excel & Visual basic were flag bearer of Low code development. Success of WYSIWYG Visual Basic & FrontPage canvas inspired a generation of development platforms. Excel concepts are the bedrock of latest low code language PowerFx.

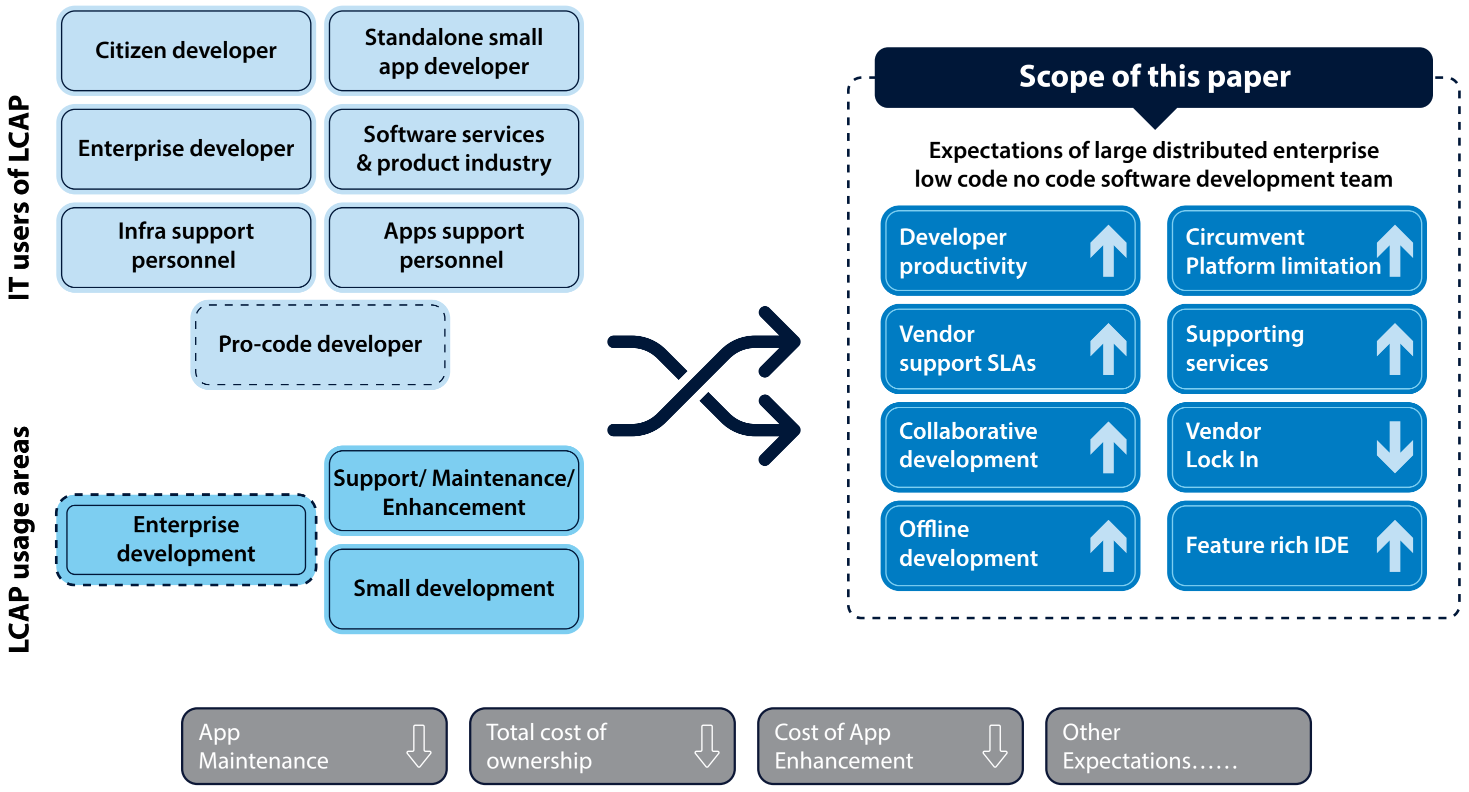
At core of LCAP disruption is the impact on development teams, resulting in process tailoring across spectrum. For example,

PROCESS TAILORING	
Traditional coding	**LCAP coding**
Detailed low level design of classes/ interfaces/ procedures is critical for developers for consistent architecture & guidance to developers	Large parts of software consists of auto-generated classes/ interfaces/ procedures, therefore developer responsibility shifted to the platform
Code reviews are intense and are integral part software process	Since most of the code is automated, not influenced by coding habits of developers, less effort is spent on code reviews

Navigating through web of expectations, long term survival of a platform depends on its ability to cater to largescale enterprise development idiosyncrasies like distributed teams, skill gaps leading to governance effort, statuary compliances, cost effectiveness and leverage existing investments.

This paper focuses on large, distributed, enterprise low code no code software development team, therefore relevant to a broad range of capabilities.

Expectations from LCAP by user types



Contemporary perspective is provided thorough Microsoft Power Platform capabilities, which is representative of industry direction, as it is placed in the leader's quadrant in October 2023 Gartner's report.

The paper does not cover combinations like development by small co-located teams, or citizen developers. It does not cover the application & infrastructure production support processes including quick-fixes and minor enhancements. It is not concerned with optimizing license costs. The author encourages a separate study of these conditions for an embracive perspective.

2. Imperatives

This section provides recommendations for LCAPs product roadmaps for meeting enterprise aspirations. These suggestions are of interest to software services organization because that is going to determine developer productivity, and by extension ROI.

Similarly, it's vital for LCAP vendors because developers observations will influence adoptions. These expectations need to be married with vendor's perception of market opportunities, and its ability to innovate & execute. Product features & roadmap are an important factor in choice of LCAPs over traditional platforms.

2.1 | Effective utilization of resources through collaborative development

Team members working in parallel on separate modules/ functionalities with subsequent integration is indispensable for efficient utilization of resources in any non-trivial engineering process. While parallel assignment of work is routine, implementing it in LCNC context needs further consideration as demonstrated below.

2.1.1 | Granular division of work

Traditionally, multiple developers code for a software product. This division of work is not so easy in a No code Low code environment, primarily because much of the interfacing code is not in control of developer, and in many cases not testable independently. Therefore, collaboration at same granular level as full coding languages is not straight forward. This also impacts modularity.

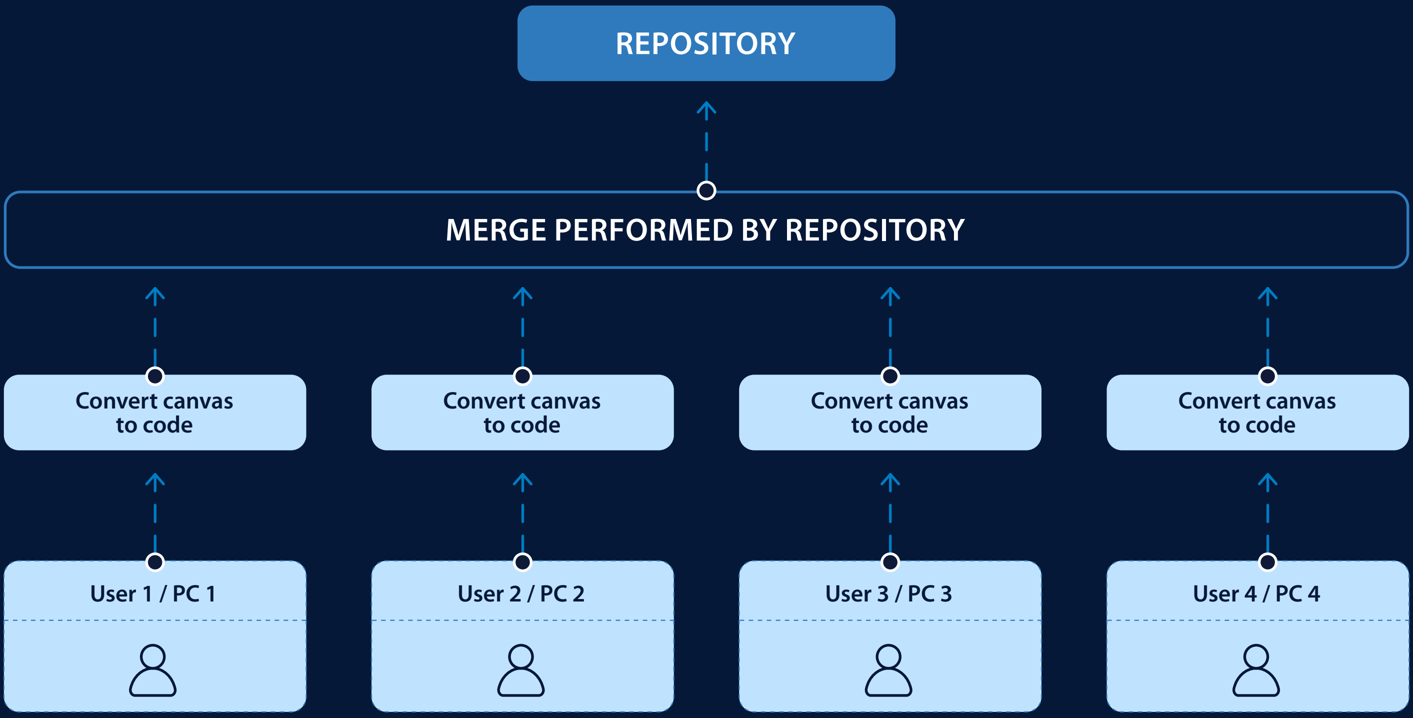
For example, In a full coding language, responsibilities can be separated by building low coupled data layer, presentation layer, and business logic. In a LCAP, taking Microsoft Power Platform as a case in point, 'Gallery' is a single control encapsulating all the above functions, therefore making it harder to separate common layers and reduce duplication.

Power Platform has attempted a degree of segregation through Power Apps Component Framework (PCF) feature and component libraries.

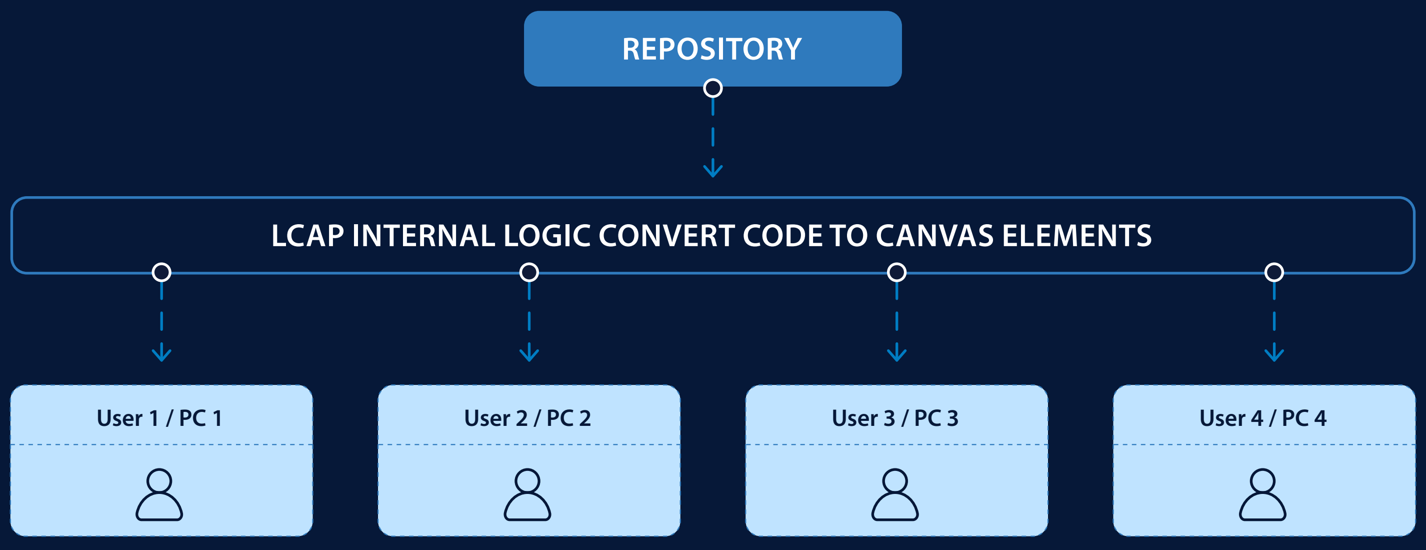
2.1.2 | App co-authoring with Version Control System

Version control systems enable division of work by allowing different team members to work on same code base. It is an essential feature of an enterprise distributed development team.

Low code development does not have 'explicit' code files. To enable effective collaboration through version control system,vendors can use their proprietary low-level architecture to transpose between automatic generated code and developer interface.



Push Request to VCS for LCAP



Pull Request from VCS for LCAP

Microsoft released Git repository integrations for Canvas Apps under 'Upcoming experimental features'. This means there is no guarantee that it will work correctly in Live projects. In fact there are reports of code overwrites, projects not coming back online in community forums. There are other work arounds, for example, Power platform tools extensions for VSCode. But that falls under the realm of pro-code development.

2.2 | User interface design:

Translating images provided by designers into code and maintain same look & feel on different resolution devices has never been perfected by UI frameworks. This is even more challenging in LCAP, as UI is created by pre-existing generic controls. Some aspects like colors, fonts are configuration led, but other elements like overall placements & alignment of UI elements are not so trivial.

As a step towards addressing this gap, Microsoft has released automation of design through import of Figma files.

2.3 | Vendor lock-in:

Vendor lock-ins adversely impact course corrections, and therefore disfavored by both software service firms, and end users. Enterprise products have long term commitments, with multiple upgrades over its service life. An option to port across vendors is crucial to de-risking these commitments.

LCAP have high vendor lock-in which is not the case with popular enterprise development languages say Java or .NET. Product companies typically resort to innovative licensing terms to foster a provider eco-system. Multiple vendors flourish in open source platforms offering choice to port software mid-stream.

At this point in time, it is difficult to imagine a scenario where a LCAP project is portable amongst any of the approx. 20 vendors. Microsoft Power platform offers import / export facilities, but it remains within the confines of same platform.

LCAP industry will need to solve this for adding muscle to the arguments over traditional coding.

2.4 | Integrated development environment:

Developer productivity is affected by the choice of IDE. Over a period, features were progressively added, blurring separation between coding tools and supporting processes. LCNC IDEs implemented transformational concepts by emphasizing centricity of visual development. LCNC IDE features & speed has re-surfaced as an important consideration. We evaluate both these parameters in below sections.

2.4.1 | IDE features:

Traditional IDEs offer features such as debugging tools, integrated ALM tools, and source control, which LCAPs need to catch up with. E.g As of now, Power platform provides only experiential features for VCS integrations. It provides feature called 'solutions' to assist in ALM & code promotion.

Similarly LCAP IDEs lack debugging tools with features like memory views, breakpoints, stop/ pause flow, what-if scenarios by changing value of variables or line of code, visible call stacks etc and Dynamic run time debuggers to debug issues like memory leaks.

As a workaround to providing rich native features, Power platform offers plug-ins for using third party traditional IDEs with its LCAP platform.

2.4.2 | IDE speed:

Due to the network response latency, exclusively online SaaS development environments do not scale up to the higher developer productivity required at times of urgency.

The issue becomes more pronounced when developers work on remote client networks using VDIs. This is due to bandwidth issues still witnessed in large parts of the world, especially offshore locations with high skillset availability. This is further exacerbated in work from home situations.

LCAPs should move towards a hybrid model, for example, latest version is available as SaaS, but option to work on local installations is available, with a way to sync up workspaces.

Microsoft Power Platform has desktop version for Automate, but other components work in a SaaS model only.

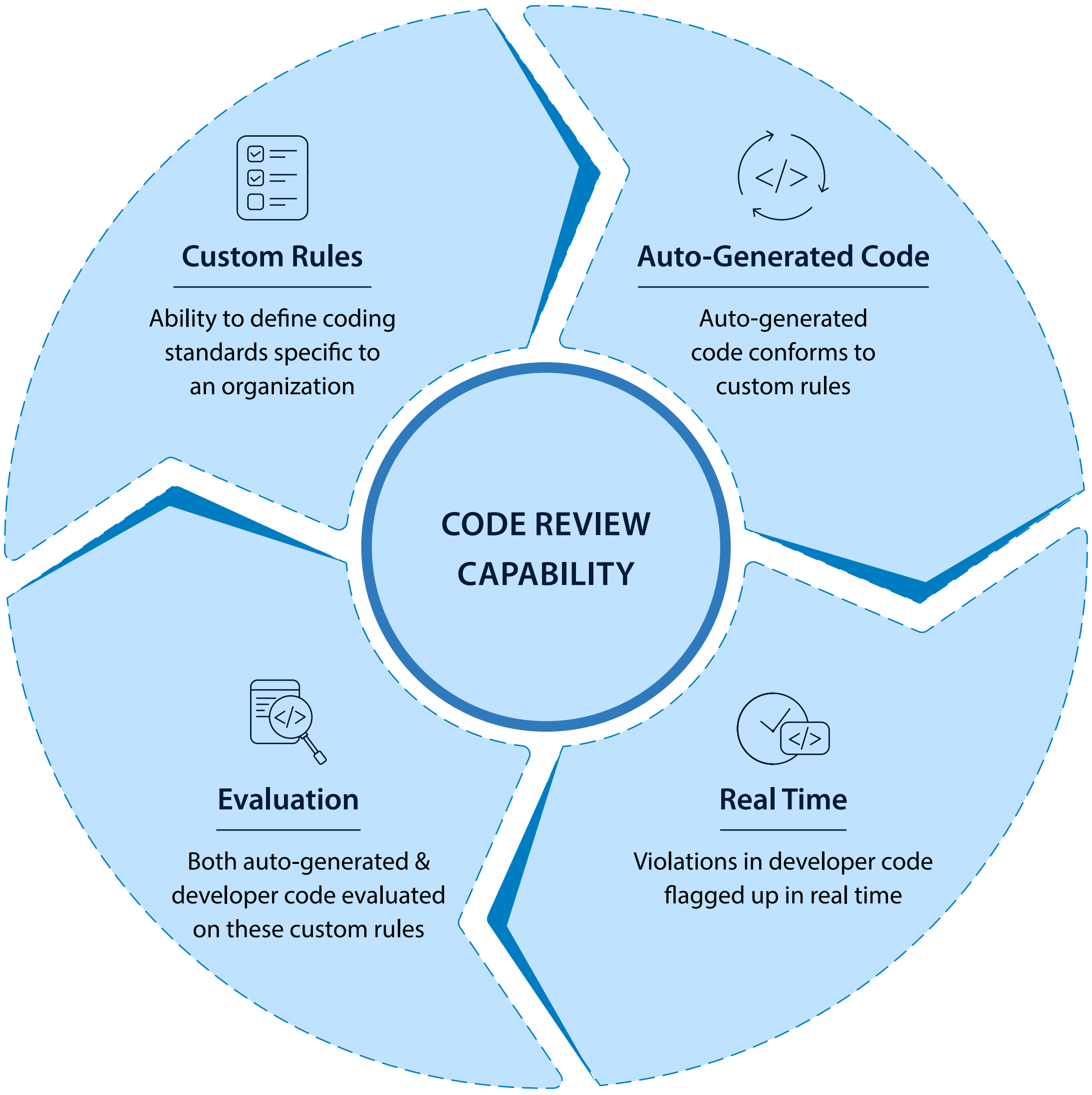
2.5 | Integrated code-review tools

Lot of developer effort has gone into adherence to coding guidelines. Industry graduated to tools-based review with organization specific custom rules & real time validations.

LCAP code is majorly automatically generated, with smattering of handwritten low code scattered in various controls. Taking these peculiarities into view, code review tools in LCAP need to further evolve & mature.

The diagram shows minimum platform capability to ensure continuity of effective code-review productivity.

Power Platform provides a separate application PowerApps code review tool for retrospective code standards review which can further enhanced to fully cover these capabilities.



Custom Rules
Ability to define coding standards specific to an organization
Auto-Generated Code
Auto-generated code conforms to custom rules
CODE REVIEW CAPABILITY
Evaluation
Both auto-generated & developer code evaluated on these custom rules
Real Time
Violations in developer code flagged up in real time

2.6 | Third-party integration & custom coding

The impact of late discovery of platform limitation during development cycle is extensive. It involves aligning multiple stake holders to reduced capability, effort to find alternatives & possible financial impact.

LCNC industry is particularly prone to it due to the inceptive nature of many features. Tools should allow circumvention through custom coding in underlying language, i.e .Net. YAML, Liquid in case of Power Platform.

For example, if a payment gateway of customer choice is not integrable through LCNC, then it should be possible through pro-code circumvention within the same platform. Similarly complex requirements like parallel threaded executions, complex relational data retrieval transformations etc need custom development.

On one hand providing extensive custom coding capabilities will go a long way in de-risking & help adoption decisions, on other it has potential to blur the line between Low code/ Pro code, impacting positioning of the product in market.

Power Platform provides ability to write pro-code through Microsoft Power Apps CLI and Power Apps component framework. It has ability to expose auto generated code to developer & integrate with standard IDEs through plugins.

3. Conclusion

While visual approach to software development has been around for many years, it has unambiguously entered the fast lane now.

This paper lists LCAP features & roadmap desired by enterprise developers. Microsoft Power Platform is used as representative of industry status. For customizations in non-trivial projects, industry will continue to have mature pro-code capabilities in the mid-term. LCNC capabilities will strengthen progressively, after which pro-code will become less significant. This trend is already visible, for example, Power platform released low code capabilities for payment gateways and low code Dataverse plugin, which were possible only through pro-code until recently.

Software development is one of the last bastions to fall to automation. Through sufficient industry discussion and debate, LCAP will become long term bets like .NET & Java. Impending vendor consolidations will further mature the industry, resulting in irrevocably changing the face of development as we know it today.

4. References

- Microsoft named a Leader in 2023 Gartner® Magic Quadrant™ for Enterprise Low-Code Application Platforms | Microsoft Power Apps
- Best Enterprise Low-Code Application Platforms Reviews 2024 | Gartner Peer Insights
- Gartner Forecasts Worldwide Low-Code Development Technologies Market to Grow 20% in 2023
- The Forrester Wave™: Low-Code Development Platforms For... | Forrester
- Official Microsoft Power Apps documentation - Power Apps | Microsoft Learn
- https://www.figma.com/

5. Glossary

ALM	Application Life cycle Management
WYSIWYG	What you see is what you get
LCAP	Low code application platforms
LCNC	Low code No code
ROI	Return on investment
WBS	Work breakdown structure
PCF	Power Apps component framework
IDE	Integrated development environment
VCS	Version Control System
VDI	Virtual desktop interface

Author details

Ashish Bansal
Principal Consultant, Infosys

Ashish is a seasoned professional in managing large program deliveries ensuring timely & quality implementations. Along with delivery management, he pursues interest in technical evolutions of Java & Power Platform, providing practitioners view & consultancy in these technologies to various ongoing projects. He has added responsibility of root cause analysis & remediation plan in case of service escalations to various engagements.

Ashish holds Bachelor of Technology in computer science & engineering from NIT Kurukshetra.

Infosys Cobalt is a set of services, solutions and platforms for enterprises to accelerate their cloud journey. It offers over 35,000 cloud assets, over 300 industry cloud solution blueprints and a thriving community of cloud business and technology practitioners to drive increased business value. With Infosys Cobalt, regulatory and security compliance, along with technical and financial governance comes baked into every solution delivered

For more information, contact askus@infosys.com

Infosys®
Navigate your next

Infosys.com | NYSE: INFY

Stay Connected