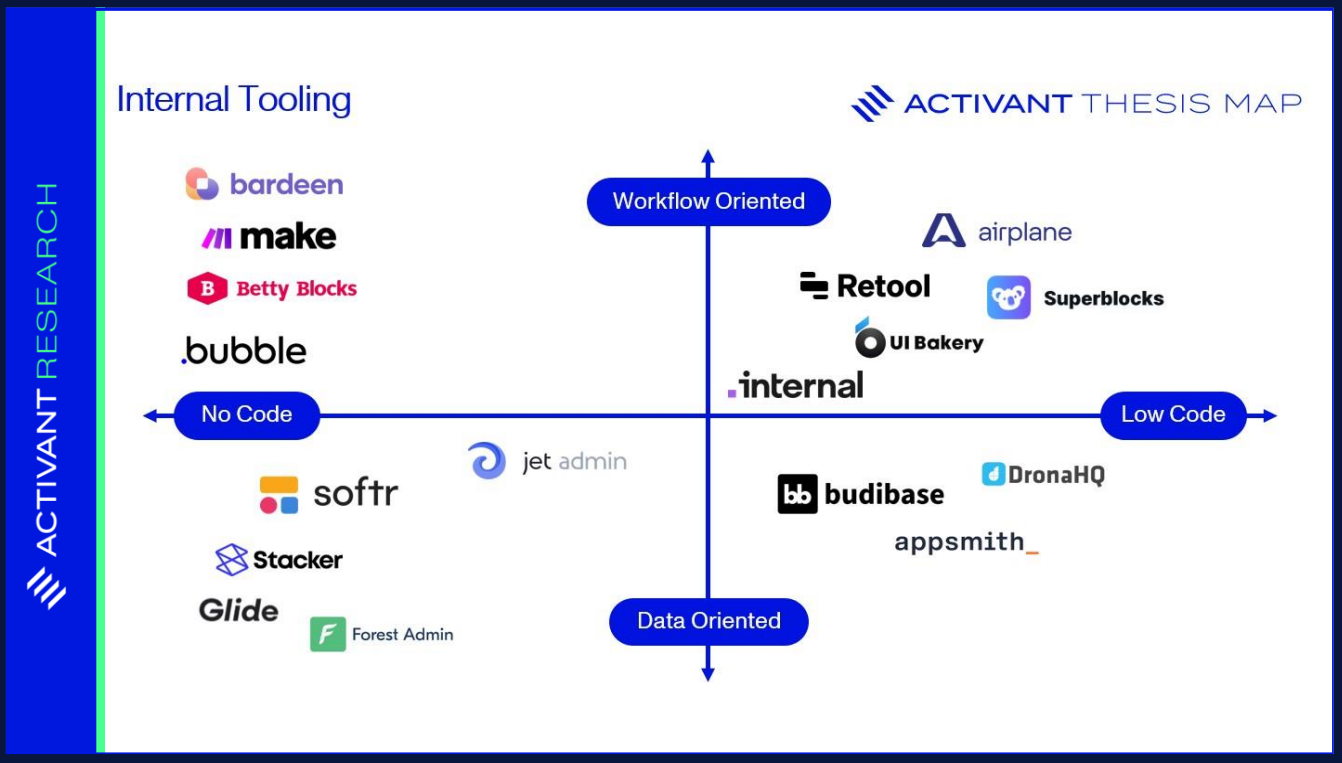
ACTIVANT RESEARCH

No-code/Low-code

Why Using Internal Tools to do More with Less Matters Today



Jono Vickery, Rebecca Rodseth, Max Thoeny

March 2023

We think that internal tooling technology has reached an inflection point. What was already a fast-growing category of software is getting more sophisticated by the day. And in today's macroeconomic environment where everyone – engineering teams included – has to do more with less, ***no-code and low-code*** tools are even more critical.

The recent Cambrian explosion of infrastructure platforms has been a boon to all builders of technology; but in today's market, it's a survival of the fittest. We believe that no-code and especially low-code platforms can now materially affect the age-old build vs buy question for startups and enterprises alike. Because what once was a clear *buy* can now become a *build*, and at lower long-term cost with more flexibility, with the use of these platforms.

And with the even more rapid rise of Large Language Models, it's clear that the way that everyone – technical and non-technical users alike – interacts with and builds software is subject to change. It's an exciting time for internal tooling – let's dive in.

What are Internal Tools?

Some problems can be perfectly solved with off-the-shelf software. Some can't. Companies spend over 30% of their developer resources building internal software[1] to solve the unique challenges that only their business faces. These challenges could be anything from exposing databases to non-technical users through analytics dashboards or creating user management portals with CRUD functionality. Teams may need to create automations for complex workflows, professionalize the execution of regular scripts or build custom applications like CRM's that fit their needs in a way that Salesforce can't.

The problem? Most internal tools are mediocre to use and even less enjoyable to build – for those that have the technical ability. Often with no clear link to revenue, internal projects are often under-resourced; the best developers want to work on customer-facing projects. User feedback is weak due to the lower user counts from users who have no alternative solutions if they're unhappy with the software. Where developers on customer-facing problems get quantitative metrics on how well every feature in their app performs, developers of internal tools might have an interview with someone from HR. Finally, the people who are closest to the underlying problems, business users, can't build the solutions that they need.

Today, we're seeing a new crop of **low-code & no-code** (LCNC) tools emerge. Focused exclusively on internal tools, these products are all about function over form. Products from companies like Airplane, Retool and Superblocks get builders to a professional, scalable solution fast with low-code while companies like Betty Blocks & Forest Admin are opening up app development to a whole new world of potential builders with no-code.

And it's about time. There are only 31.1 million trained software developers, amounting to a global shortage of 1.4 million, which is set to rise to 4 million by 2025.[2,3] Firms that we've spoken to have seen an 80% reduction in the time taken to ship a product developed with low code. Meanwhile,

developers amount to less than 3% of total global knowledge workers.[4] One can only imagine the possibilities of bringing the remaining 97% into software development.

These tools promise to speed up building things like customer dashboards, but we have also spoken to younger start-ups who are building their entire software infrastructure on LCNC for absolute control.

We are seeing the early signs of mass adoption of no-code/low-code development tools for internal software.

The End of ZIRP and Beginning of New Build or Buy Decisions

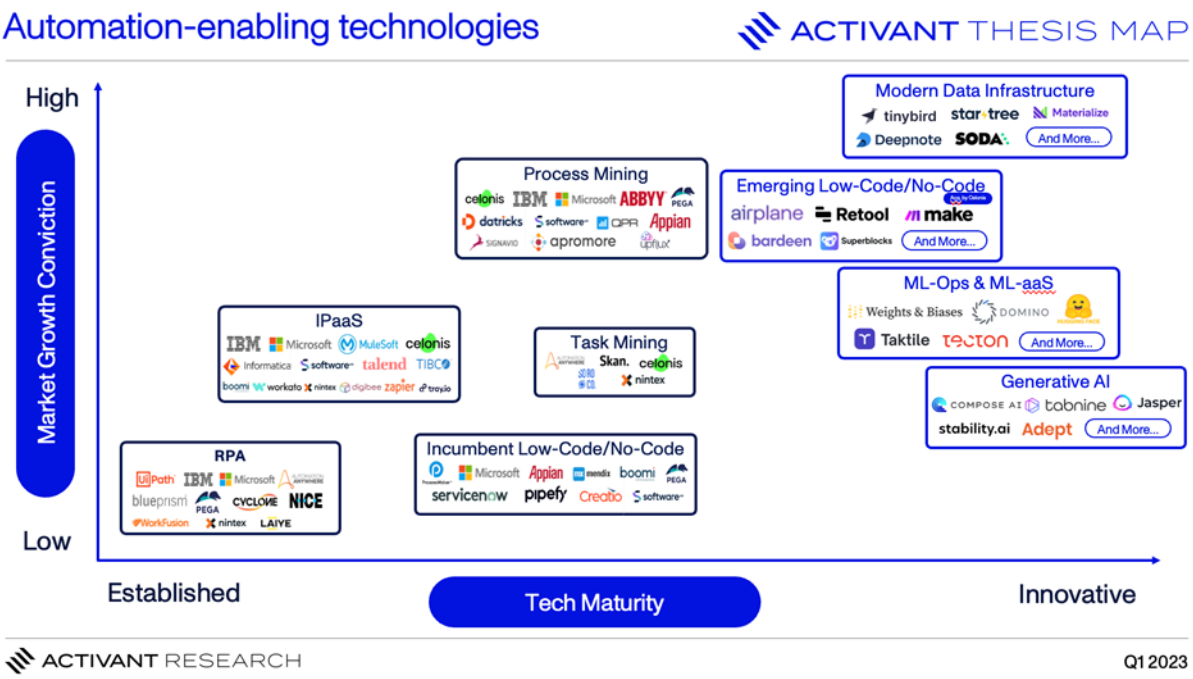
Where interest rates will be a year from now is anyone's guess, but one thing is clear – the period of excess is over. Its time to get fit and many start-ups are now serious about getting burn under control. Every expense on the income statement is going to come under scrutiny. We've discussed the ever-expanding the number of apps that companies use, and this could be an area for some easy cuts. There is definitely some low hanging fruit here but we believe that the existence of low-code tools fundamentally changes the build vs buy decision. If you can cut build time by 80% with the advent of low-code tools like Airplane, Retool or Superblocks, it suddenly becomes *a lot* more attractive to build internally vs buying the next flashy SaaS tool.

Take Meta and their "Year of Efficiency." In Mark Zuckerberg's most recent update, he talked about how Meta is *"investing in tools that will make us most effective over many years, not just this year — whether that's building AI tools to help engineers write better code faster, enabling us to automate workloads over time, or identifying obsolete processes that we can phase out."*

He cites that their new open source compiler, Buck2, works 50% faster, which meant that: *"an analysis found that engineers using Buck2 often produced meaningfully more code"*

And as infrastructure investors, this is something that we care deeply about. While the infrastructure layer of any tech stack can be a great place to be, many early-stage infrastructure businesses face graduation risk – the point where it's cheaper for your customers to build the solution for themselves. Again, the advent of low-code might drastically change the economics of the build vs buy decision. *There could be large ripple effects through broader software markets.*

Revisiting the Thesis Map



Revisting the Activant Thesis Map we presented in our Hyperautomation report, we see internally focused LCNC tools as a deeply innovative and high-growth emerging area.

Generative AI is undoubtedly the most innovative technology that we have seen in the recent past. Large language models are exploding, both in their computing capacity requirements and their capabilities. The most well-known Generative AI application is ChatGPT, whose model is based on GPT-3, a 175 billion parameter machine learning model that is guestimated to have cost $10mn - $20mn to train (exact figures are unknown).[5] However, it is unclear where the competitive advantage lies, and if it is in sheer scale or distribution advantages, then the benefits of this AI wave are likely to be captured by existing mega-cap technology companies. Ultimately, we see a large degree of variation in potential outcomes here.

5) Andrey Kurenkov, Last Week in AI, GPT-3 is No Longer the Only Game in Town

The growth of modern data infrastructure is more certain. Humanity continues to grow the creation of data at exponential rates and simultaneously place ever greater requirements for the operationalization of that data, in real-time. Another infrastructure play, **ML-Ops & ML-aaS**, are providing exciting tools for machine learning engineers to enhance productivity & improve model explainability. While the tools inherent in providing a visual interface to produce code are slightly less innovative and more deeply wedged into existing workflows, we see this as a key growth area going forward, for all of the reasons already discussed.

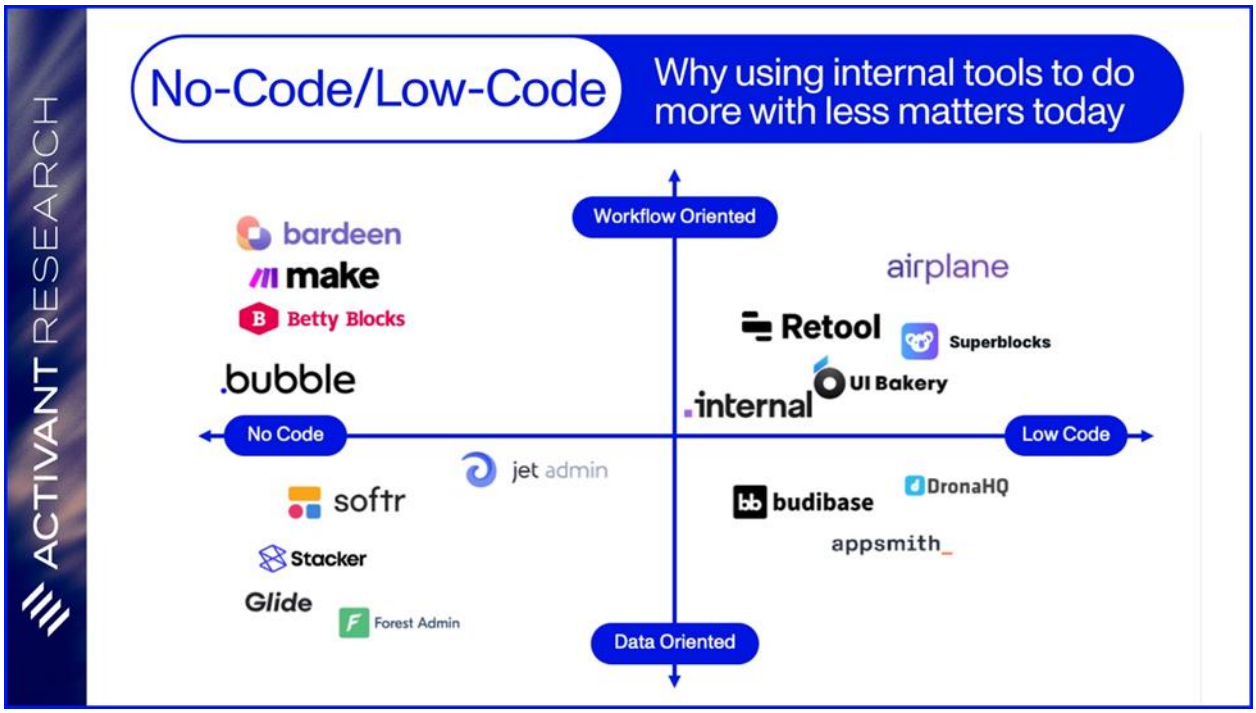
Zooming in on No-Code & Low-Code

No-code tools allow *anyone* to develop software by using out-of-the-box templates, pre-built UI components and API connectors. Development is done in intuitive visual environments with the principles of point and click & drag and drop. Essentially, no-code abstracts away the complexity of code and presents it to the user in the form of a graphical user interface. True to the name, anyone in the organization can build software without writing a single line of code.

Low-code tools are aimed at developers, who can augment their traditional coding using the principles of no-code to rapidly speed up development. Low-leverage activities are powered by the low-code environment so that developers can focus on their highest-leverage activities. Fundamentally, low-code enables developers to *write* less code, but *produce* more.

	Low-Code *Making developers lives easier by augmenting code with no-code features*	**No-Code** *Enable anyone to build using intuitive, visual features like drag and drop and pre-built UI elements*
Workflow-Oriented *Focused on automating tasks and executing functions*	**Developers** write a program to do run scheduled tasks or automate all of the flows associated with a user sign-on	**Marketing** team builds a tool to connect a number of applications and automate marketing campaign
Data-Oriented *Create an app with CRUD access to company databases*	**Data Scientists** could spin up a data scraping tool, clean data and deliver insights in an analytics dashboard	**Operations** teams can build a dashboards to track live company metrics and enter data directly into company databases

Activant Market Map



Employees at LCNC start-ups have made it clear to us, the choice to be either low-code or no-code is a critical philosophical decision. This early positioning creates organizational ripples from go-to-market strategy to company culture.

Another key defining trait in this market is whether tools focus on powering dashboards and data visualization (data-oriented) or automating workflows and scheduled tasks (workflow-oriented).

Reading the Map

As we see it, the no-code vs low-code and data vs workflow orientation becomes a key differentiator across the market. Software buyers should be sure of exactly what they want to achieve and who their users are.

- Tools like Forest Admin, on the no-code, data-oriented end of the spectrum, should find success within their niche, by giving non-technical users in the organization the ability to monitor data that is stored in databases which those users would otherwise be locked out of without learning SQL.
- Similarly, tools like Betty Blocks will unlock app development for the whole organization to automate and manage workflows like service requests, risk assessments and document reviews.
- Low-code, data-oriented solutions like Appsmith will allow developers to build sophisticated admin panels and dashboards with access to key database & data warehouse integrations like PostgreSQL, MySQL and Snowflake.

- Workflow tools like Airplane, for example, have made it very clear that they are geared toward developers. While the company does provide UIs for classic dashboards, their key value proposition is to professionalize workflows being run with cron jobs and ad-hoc scripts.

Key Features and Considerations

Enterprise-readiness

For these products to be fit for the enterprise, they need to be delivered within the context of quality features primarily in security and compliance. We have seen a large portion of the landscape adopt single-sign on, granular permissions management and audit logging. Airplane have built request and approval flows into the product, while also being deployable on premise, often a key consideration for customers regulated by GDPR.

Enhanced developer features

Superblocks appear to have pulled ahead with enhanced features like version control with git, automated testing and observability integrations. Built to bring developers and product managers together into one tool, companies like Dynaboard lead the way with collaboration features. A common issue with low-code apps has been the ability to debug code that is hidden behind pre-built UI's. Retool and Appsmith have built debugging and troubleshooting tools into their editors.

Extensibility

There is an inherent trade-off between extensibility and accessibility. No-code tools like Glide provide a huge library of templates to allow business users to build software in minutes. But apps can become limited by available templates and components. Many companies extend functionality by allowing code-based editing, such as Superblocks who support front-end editing with Javascript and back-end editing with any language. Airplane's approach to this problem is noteworthy - their product sacrifices accessibility by being accessed as code, but by embedding itself within their customer's codebase, removes the limits of what developers can build.

Pricing

Many tools provide seat-based pricing at the lower tiers with a freemium model. The freemium model is highly favorable for driving growth through testing and iterating. However, seat based pricing can become prohibitive as the number of users scaled and potentially difficult to validate for simple use cases making a few database queries.

What's Next?

Large Language Models

We started writing this piece last fall, and the extent to which the conversation around LLMs has changed has been astonishing. The internet is buzzing on chatGPT and its ability to assist programmers, while the Github's Co-Pilot now turns natural language prompts into code. Its suggested that co-pilot is writing up to 40% of the code for its users. Low-code/no-code has the explicit goal of rapidly speeding up application development, and we expect to see the application of AI to this space to accelerate rapidly.

Essentially, *LLMs may radically change computing*. LCNC presents users with a graphical user interface, which is how the vast majority of humans interact with machines, but a natural language interface (i.e., a chat conversation) might be a far superior model. Integrating tools as powerful as co-pilot into existing LCNC tools might be an immense accelerator. Further, it could be just the step change in capabilities that will allow no-code software development to have a moment.

Speaking at Berkeley last month Nvidia CEO Jensen Huang said: *"For the last 40 years, we have made computers harder and harder for people to program and that's why the technology divide has been so large and the technology divide is getting larger and larger except till 1 day, all of a sudden, everyone can program a computer. You just have to prompt this thing to write a program for you, do something for you, automate something for you."*We wholeheartedly agree – but take it with a grain of salt coming from a key supplier that stands to benefit from the coming AI wave.

Its Low-Code, Not No-Code (Today)

The promise of bringing one billion knowledge workers into the software development flow would represent a complete phase transition in software markets – but we're still a ways out. While no-code users are extracting immense value from these tools, they don't have the capability to transform the market. Product managers that we have spoken to note that the primary issues that these tools run into is the fact that they are *too opinionated*. As we dig into later, there is an inherent trade-off between ease of use and extensibility. Today, that is a major hurdle that no-code is bumping into. As the market is structured today, it's really low-code that promises to be an exciting area for the future.

Breaking in

Based on our conversation with spoken to product managers, who we believe that the world is *absolutely* going to move in the direction of LCNC, but a few factors limit this. Where an organization has already built the majority of their software, it is tough to implement low-code/no-code tools that live outside of the codebase, which would require building everything from scratch. Developers tend to love using low-code tools, but can only deeply embed them into organizational development flows with the buy-in of product managers. Without overcoming these hurdles,

software built with LCNC techniques tends to live in siloes, being used only for a few discrete tasks. A paradigm where the world's software is built on LCNC, will require deep organizational buy-in.

Openness, and the extensibility-accessibility compendium

With some LCNC tools living in siloes outside of the organization's core development flows, its clear that there may be a big opportunity for companies like Airplane who can operate inside of the companies codebase. Of course, this shift away from accessibility may limit the core value proposition of low-code. Going forward, companies will need to find ways to stay true to their ease-of-use promise while becoming more deeply embedded into development flows.

Going external

As these low-code/no-code software tools look to become more deeply embedded into development flows and drive their next wave of growth, they will inevitably need to satisfy externally facing use cases. These transitions will not be simple to navigate. Core issues that companies will face include scalability and form factors. Infrastructure will need to be robust and elastic, moving from tens of internal users to tens of millions of external users and clunky tools that get the job done will need to shift to providing rich user experiences.

If you're working on a company to make engineers more effective and knowledge workers more like engineers - we want to talk.

Endnotes

1) Retool, The State of Internal Tools, 2022

2) Developer Nation, Pulse Report 2022

3) IDC, Quantifying the Worldwide Shortage of Full-Time Developers

4) Gartner, 2019: When We Exceeded 1 Billion Knowledge Workers