Low Code Ops

And The Art Of Abstraction

Figure 1: Layers of abstraction in an ever-changing environment. Photo by PhotoHolic.

Written by

Isak Hassbring & Gustaf Lidfeldt

DD2482 Automated Testing and DevOps
KTH Royal Institute of Technology
Stockholm, Sweden
April 2021

Contents

1 Introduction

A lot is going on in the software engineering world. The recent years have shown tremendous improvements within the fields of development and operations - following the increasing popularity of *DevOps*. DevOps in general could be seen as a set of practices serving as abstraction layers that aim to let more people be more effective, by building interfaces and automation around difficulties - hence abstracting them away [?]. The same goes for many engineering fields, e.g. computer engineering is an abstraction on top of electrical engineering. Programming languages are abstractions (sometimes even *on* abstractions) on machine code [?]. Among programmers, a “one-liner“ is often considered superior compared to multi-line programs (given equal task and performance). Why? Because humans love the art of simplicity and abstraction. It is as beautiful and satisfying as it is easy to understand [?] [?].

What is the next generation, i.e. the next layer of abstraction? It could be argued that the next generation of development practices, is perhaps currently emerging as the hugely disruptive *low-code/no-code* platforms ct. In the low-code/no-code space, the spreadsheet program Microsoft Excel could be seen as the birth of the practice around 1990, as it lets the average person compute calculations without any programming skills [?]. Now, a lot has happened since then and it in recent years low-code/no-code has emerged as an automation pool to count on. Today, platforms such as Peltarion.ai even go beyond the once-beloved one-liners and let creators (people building applications) set up, train, test, and deploy cutting-edge deep learning techniques with or without needing to write code. Bubble.io and lets you write auto-generated interactive front ends, also as a no-code practice. Microsoft is investing heavily in its low-code suite PowerApps, that provides automation of multiple business processes and user interfaces [?] [?].

As an example, we all know what the emergence of the no-code platform wordpress.org meant for web developers as it completely reshaped their industry. So, what does the broad no-code emergence mean for teams in both development and operations? Is this the beginning of the end of DevOps as we know it? We will explain why this matters for a DevOps engineer (or any engineer in general!) and discuss possible practical implications of low-code in the future.

2 Background

2.1 Low-code and the art of abstraction

Low-code and no-code development refer to the usage of a GUI (graphical user interface) for code generation instead of typing code by hand within an IDE (integrated development environment). Low-code development typically follows an agile methodology such as Rapid Application Development. Agile and DevOps are both designed to achieve faster development and deployment. [?]

While low-code platforms are not (and will perhaps never be) useful in all development scenarios, they are increasingly useful for a large and growing set of applications and practices. A UI/UX designer can for instance use *Figma* [?] to automatically generate fully functional React-JS components from their design, for prototyping and user testing. This hugely speeds up and automates the front-end application development. It lets the front-end developer take a step back and focus on API:s, integration, functionality, putting the pieces together instead of building the views. This is a more efficient use of resources and increases productivity.

Low-code platforms are especially useful in smaller, agile organizations or for smaller projects with few development resources. Low-code tools fit very well in such application development life-cycles as it lets programmers, as well as non-programmers, create custom-made apps or services for specific use-cases - faster and easier [?].

2.2 DevOps perspective

What serves as a good area to provide a low-code platform framework? Standardized development practices could be a perfect scenario. Why? Well, if something is standardized, it typically involves repetitive tasks. Hence you could build reusable, modular, object-oriented code around it. And if so - you'd be able to provide a GUI for it as well. And in the scenario of a standardized end-to-end independent task in development or operations, the entire task development could be made through a low-code GUI.

Now, what's interesting here is that standardized dev/ops practices are oftentimes *exactly* what DevOps provides development and operation teams with, and hence a subset of these practices can be provided with end-to-end GUI:s. This is in fact what several companies have realized, and a lot of big players are entering the field one way or another. *All* the DevOps cloud tools provide solutions that *do* fit into the low-code definition. Many tools still long way to go and researchers are looking into the field [?]. For us to sneak a peek into the future, we will look at the ones that are one step ahead - such as Peltarion and Figma, in the next section.

3 Practical examples

In this section, we will cover some concrete practical examples of disruptive low-code tools.

3.1 Peltarion - a low-code platform perfect for ML Ops

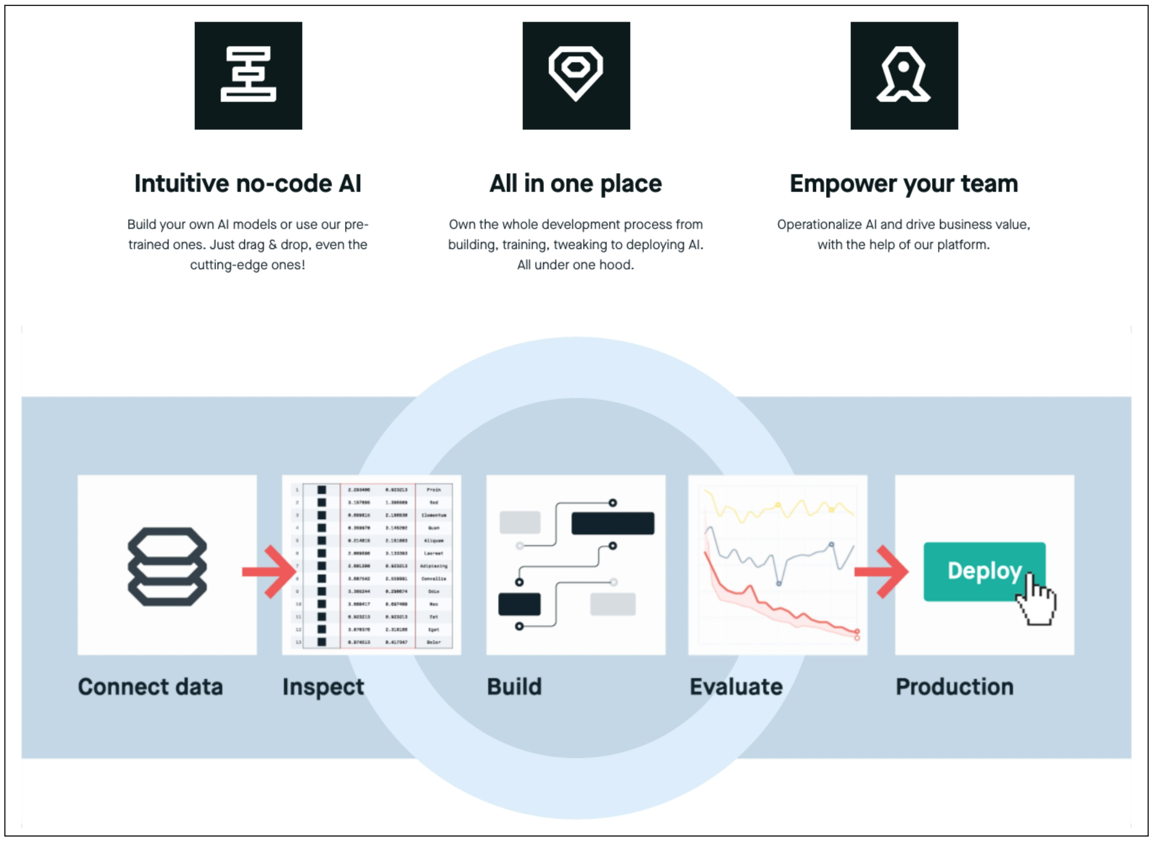
Intuitive no-code AI
Build your own AI models or use our pre-trained ones. Just drag & drop, even the cutting-edge ones!
All in one place
Own the whole development process from building, training, tweaking to deploying AI. All under one hood.
Empower your team
Operationalize AI and drive business value, with the help of our platform.
Deploy
Connect data
Inspect
Build
Evaluate
Production

Figure 2: ML Ops enabled by Peltarion. Photo by Peltarion.

Peltarion [?] is a Wallenberg and EQT backed Swedish company that provides an end-to-end machine learning (ML) platform that makes the fuzz unfuzzy when it comes to ML usage and ML Ops. Deeply technical companies such as Tesla, Spotify, King, etc are using the Peltarion platform. Peltarion offers a no-code platform where you can use their cutting-edge pre-trained AI-models or build your own ones with a drag and drop interface. The entire development process from building and training to deployment is held in the cloud on Peltarion's platform. A tutorial is available here.

We have tried it - here's how it works:

1. First, you create a project. A project holds the pre-processing of datasets, model building, evaluation, and deployment. It's configurable as a shared resource and can be used with others.

2. Then, you're asked how you would like to provide your model with data. There are multiple ways to connect to data: through an API, import data from data-warehouses such as BigQuery or Azure Synapse, local upload, URL import, and perhaps the coolest alternative: use Peltarion's

library of free pre-processed datasets - a huge offload for a large amount of ML applications. See figure below.

3. Next, the GUI lets the user inspect and set up the data thoroughly.

4. Now it is time to define how to build and use the pre-trained model. When ready, one simply presses *Run* - and the model starts running.

5. By pressing *Run* in step 4, we move over to the Evaluation section. Here, we're able to visually follow along during the model run and evaluate its performance in real time. The platform then automatically prepares the best-performing epoch for deployment.

6. Finally, literally by pressing *Deploy*, the model is deployed to production, directly accessible through an auto-generated API endpoint.

For anyone familiar with more conventional ways of handling ML development processes and operations, this is a huge step indeed. As far as abstraction and simplicity goes, the Peltarion platform makes it much easier to work with ML Ops, and it is doing so without requirements on coding skills.

3.2 Figma - a low-code tool that changed front-end development forever

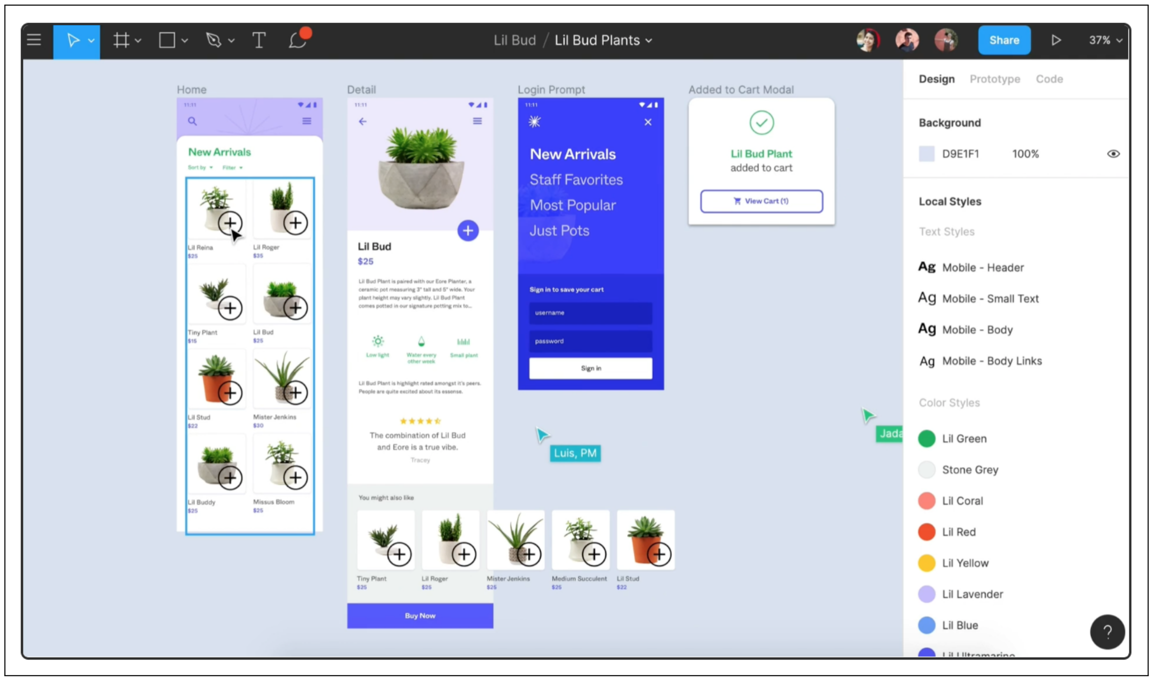

Figure 3: Demonstration of Figma's low-code platform for front-end. Photo by Figma.

Figma is a popular low-code tool used within front-end development and design teams to rapidly speed up development phases including GUI parts. Figma lets a designer or front-end engineer design an element or a component just as you would do it in a traditional design tool - but instead of generating an image, your result is fully reusable JavaScript components (e.g. React-components), which can be used directly in prototyping. The use of Figma speeds up the actual code generation and enables people who typically are not considered programmers, to participate in front-end development or code generation. As compared to traditional hand-coding, this is a big change [?].

3.3 There are many more

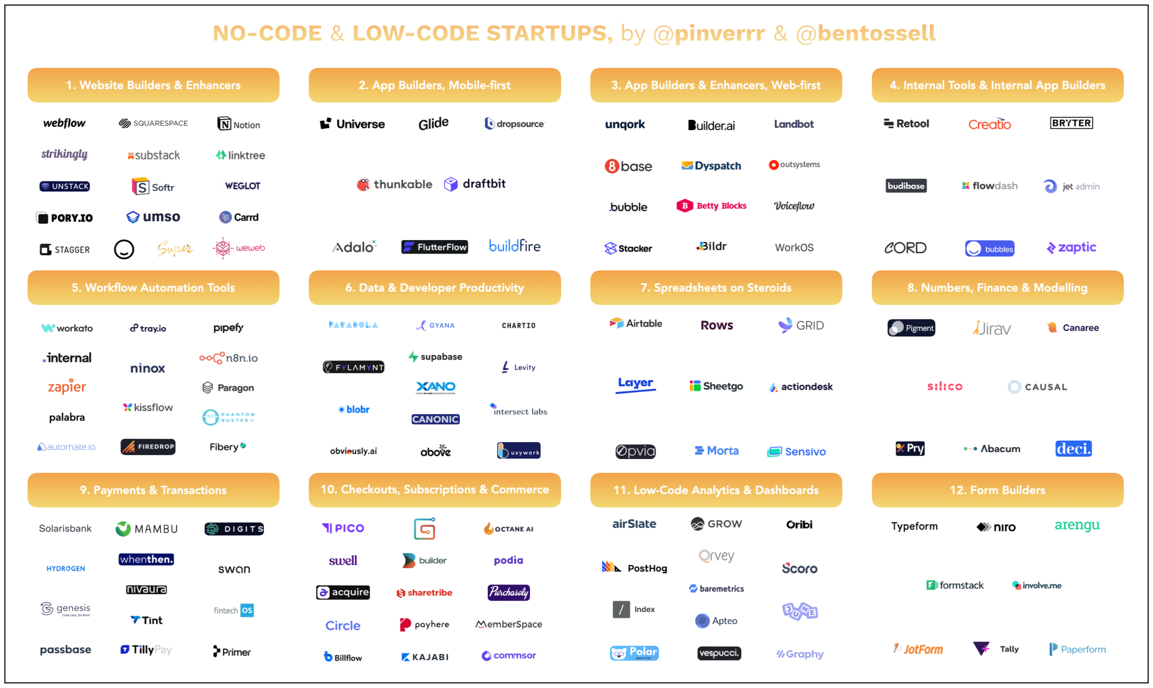
NO-CODE & LOW-CODE STARTUPS, by @pinverrr & @bentossell
1. Website Builders & Enhancers
2. App Builders, Mobile-first
3. App Builders & Enhancers, Web-first
4. Internal Tools & Internal App Builders
5. Workflow Automation Tools
6. Data & Developer Productivity
7. Spreadsheets on Steroids
8. Numbers, Finance & Modelling
9. Payments & Transactions
10. Checkouts, Subscriptions & Commerce
11. Low-Code Analytics & Dashboards
12. Form Builders

Figure 4: Examples of no-code and low-code platforms. Illustration by Pietro Invernizzi.

There are numerous low-code tools to check out, and we can't cover them all within this essay. For the curious reader, please check out the image above or some of our favorites: Substack, FlutterFlow, Workato and Chartio, or try to understand the space better via e.g. Sahay et. al [?] or Invernizzi [?].

4 Reflection

4.1 Defining the what and the why of Low code Ops

We can define Low-code Ops as the practice of including *both* code-centered development tools and processes as well as low-code tools, in the development and operations life cycle. The difference between DevOps and Low-code Ops is that DevOps streamlines everything almost like an hourglass, through the use of *code*. Simply put, everything DevOps is strongly related to code. What if there is no code involved in parts of the workflow? Suddenly it would be a struggle to put git be the core of the work processes. This is a struggle for DevOps today that must be addressed to keep up with the pace of low code, as more and more people will be able to participate in what is typically seen as development processes. Furthermore, *modeling* has been found lowering the learning curve of DevOps processes for for non-technical users enabled by low-code platforms. Colantoni et al. propose a conceptual framework for modeling and combining DevOps processes and platforms, as a future solution [?].

5 Limitations

5.1 Abstraction comes at a price

Testing is difficult as awareness, visibility, and control decreases. A graphical user interface that allows a developer to drag and drop different functionality is indisputable a time saver. Without full control of what is abstracted, testing and performance control could be an issue. There are also risks connected to scalability [?]. Organizations should recognize that there are many good reasons why it is costly to develop software. Qualified experienced developers are costly, but if an organization takes shortcuts they might spend more on re-writing bad code (auto-generated or not) when they could have just writing good code from the beginning [?].

5.2 Source code ownership

The biggest no-code platforms all state that the data you generate is your own. However, for no-code platforms, you generally do not own the source code of your creation. This results in you having to rebuild your entire project in case you wish to not use the platform anymore. No-code platforms can be great if you quickly and cheaply want to develop a minimal viable product to help raise funds for your startup. But further development might be limited by the number of plugins the platform has. Furthermore, if the value of your company is based on an algorithm and you do not own the ownership of said source code potential investors would be reluctant to invest in you. Bubble.io addresses some of these concerns by saying that you own your design and in the case that you wish to export your project, they can provide a "raw dump of your application logic and visual design in Json"[?].

6 Conclusion

The essential goal of all the cultural shifts in tech, is higher efficiency [?]. While DevOps enables organizations to speed up the delivery of code it is limited in its impact on the speed of code production. Continuous integration and deployment are great, but a system is only as fast as its slowest part - actual code commits. An organization might need to build software quickly for internal or external use, but oftentimes it is too expensive to hire software engineers. Here low-code development could play a crucial part. With low-code, novel developers can join along without the complex lines of code, while the highly skilled developers can focus on core functionality.

Is this the end of DevOps? No, it is not. Low-code is still dependent on traditional development and operations teams enabling the underlying infrastructure [?]. But, a growing set of what is typically considered DevOps practices will probably not require the DevOps engineer expertise in the future, due to low-code platforms providing abstraction. And for the DevOps practice itself, it *could be* a good idea to try to emerge from the dependency on services such as git being the core foundation of DevOps, and try to find a way to better include or collaborate with low-code development practices. For instance, such a solution could be to come up with a tool that can let organizations manage and work with version control not only for pure code files but also for progress and development made in low-code tools. Nevertheless, the need for traditional coding languages is still and will still be critical in restricted areas for a while. All the existing no-code platforms come with their respective limitations and developers will still need to fill those gaps. However, low- and no-code platforms provide smaller companies with limited resources to quickly throw together an application that fulfills certain requirements.