GIGAOM

REPORT

Real-World Citizen Development v1.0

How To Achieve Sustainable Rapid Delivery

JON COLLINS | FEB 22, 2021 - 9:53 PM CST

GIGAOM

Real-World Citizen Development

How To Achieve Sustainable Rapid Delivery

TABLE OF CONTENTS

1. Summary

We are told that low-code and no-code models offer an opportunity to deliver software much faster, and enable less technical and business-facing individuals—citizen developers—to participate in application creation. While there are many benefits to this, how can it be done such that short-term gain doesn't mean longer-term pain?

2. Introduction

Who can forget the reworking of *The Sorcerer's Apprentice in Disney's 1940 classic Fantasia*? In this salutary tale, Mickey Mouse discovers to his joy that he can make two brooms out of one, then three, then four, and so on. His smiles quickly lead to panic when he realizes that he cannot control how many brooms are created, nor what to do with them.

And now, we have citizen developers, who can harness the power of low-code and deliver innovation to their organizations. The idea of citizen developers, and their relationship with low-code in all its flavors, is not new, indeed it's been on the board for at least five years. So, what's changed?

Ultimately, like so many technology categories, low-code has reached the point where it needs to scale to deliver on its broader potential. Citizen development cannot mean uncontrolled development for a number of reasons, not least is that (as Mickey Mouse so neatly illustrates) uncontrolled anything leads to chaos.

Thinking more broadly, fears around such chaos can also get in the way of innovation as executive decision-makers are reticent when it comes to adopting low-code approaches, or empowering their people to deliver innovative solutions.

The response, as we will see in this paper, is one of balance between enablement and good practice. In this paper, aimed at technology leaders, business and process analysts, operational managers, and indeed, those looking to innovate and become citizen developers themselves, we consider the role of citizen development. We also aim to set out how to deliver on its potential in a way that marries shorter-term success with sustainability.

We cover:

- The relationship between low code and citizen development
- Core principles of sustainable rapid delivery
- Steps to building a citizen development practice

In conclusion, we set out how organizations can ready themselves for low code in general, and citizen development in particular, in a way that maximizes effectiveness without increasing business risk.

3. Why Citizen Development?

To answer this, we first need to ask: what is low code, anyway? The clue is in the name: it is about taking away challenges caused by what we might call 'High Code'. Both traditional and modern software development require technical skills across areas including:

- Writing the code itself, learning how to program and produce elegant, maintainable software
- Integrating with third-party systems and services. For example, via APIs each of which can be complex
- Building in capabilities around security, data privacy, and compliance

Low code aims to address these areas and as a result, lowers the skills barrier to application building, enabling the citizen developers we discuss below. Different low-code solutions can address different scenarios. For example, they may be:

- aimed at developers, or less technical business users;
- forms-based, or follow more of a process-modeling approach;
- focused on delivering web-based, or mobile apps, or a combination;
- better suit smaller or larger organizations.

While it is worth choosing a solution based on an organization's own needs, all solutions share the drive to lower the barrier to innovation and provide a functionally-rich platform that minimizes the need for prior experience from its users. This means that less-technically skilled stakeholders can take the baton to deliver highly-capable software to their businesses, while the organization as a whole can accelerate its use of software to drive and differentiate its business.

So-called citizen developers can:

- Create working applications quickly and directly, without having to describe their needs to technical teams (which adds time and risk)
- Engage with technical teams on solution delivery; for example, creating front-end functionality built upon existing capabilities
- Build short-life solutions to immediate needs; for example, creating a dashboard or a set of forms for tracking inventory or doing quality checks
- Take responsibility for feature roadmaps according to their own "customer" needs, which may be internal or external

- Deliver on priority application needs, which might be subject to delays if they are requested from the IT department

By empowering citizen developers, low code helps achieve an organization's digital transformation goals. However, the reduced barrier to entry does not mean that governance is thrown out. All software is complex and requires governance to manage this complexity. So, how can we apply governance to citizen development?

The good news is that core principles of software delivery best practice can be applied to low-code environments, without damaging the benefits of citizen development.

We look at these below.

4. How to Achieve Sustainable, Well-Governed Rapid Delivery

To determine what needs to be in place in terms of systems, processes, and procedures, so that the organization can empower citizen developers and keep control, we can draw on some founding principles of software development practice, learned from the experience of managing complex projects.

Particular governance areas that map onto low code and can be adopted by citizen developers include:

- Change Management
- Configuration Management
- Requirements Management
- Compliance Management
- Quality Management

Let's look at how these can map onto low code/citizen development practices.

Change management goes to the heart of good innovation practice. The essential goal of innovation is to create and deliver something new: while creation may happen quickly, delivery can struggle. Change management involves activities around managing individual deliveries, together with who they are for and what they need to succeed.

From a citizen development perspective, even the simplest of applications can be considered as an initial delivery and a series of updates, for example, based on feedback from its users. This information can be managed as a change log. The outset the log can be a simple journal or spreadsheet containing the date of a requested change, who requested it, what it entails, whether it has been dealt with, and when. It is also useful to give each a unique number or identifier.

If the application becomes more complex, with multiple changes to be considered at a time, the citizen developer can work on the basis of releases and decide what changes should be incorporated in each. If these need to be planned ahead, then the developer can create a roadmap to allocate changes to planned releases. Note that if a spreadsheet is becoming too cumbersome, you could consider a simple low code application to manage application delivery (see: *Use low code to manage low code*, below).

Configuration management is the art of keeping tabs on individual elements of the low-code application and how they evolve over time. Just as program code can be managed using a version

control system, it is also worth it to keep time-stamped snapshots of a low-code application and store them somewhere safe.

Configuration management also applies to changes in how the application is built, where it is deployed, and any other dependencies. For example, the citizen developer may need to log:

- Data models and how they relate—for example, a person has an address and other information associated, which can grow over time
- Devices that the application has been tested on, their operating system versions and characteristics
- Low-code development tools and their versions, as these may evolve and add or change functionality
- Services, data sources, and interfaces used, for example, logging the API version required
- Infrastructure and connectivity, for example, whether the application relies on a particular cloud-based data streaming service.

Again, a simple log or spreadsheet would suffice in the first instance. Such a log offers a great deal of support when it comes to referencing how the application, and its dependent resources, have changed over time. The fundamental goal is to be able to say, for example, "Let's go back to that configuration we had last September." By doing so, citizen developers can also define versions for specific groups, for example, different clients, or accessing different functionality.

Requirements management extends the idea of change management to the functions and non-functional aspects of the application, particularly if the application is to take into account multiple needs.

In a simpler scenario, a citizen developer will just build something and learn about what is needed as they go, managing any changes as they arise. However, what if, say, application needs are defined by multiple people, collaboratively?

For example, the citizen developer may undertake, instigate, or learn from interviews, workshops, or other collaborative requirements-gathering exercises. In these cases, somebody will need to manage all the ideas coming in, log them as requirements, prioritize how they are going to be dealt with, and potentially report back.

This means they need to be managed in some way; for example, in (you got it) a log or spreadsheet. Once requirements have been confirmed and prioritized, they can feed into the change management process.

Compliance management doesn't go away for low-code applications. Regulations are sets of rules that need to apply to data and how it is processed; in general, any application that needs to manage

data about people will be subject to multiple regulations. Software may also need to conform to rules around health and safety, due diligence, or other criteria, depending on what it is for.

Right at the outset of creating a low-code application, citizen developers can consider whether the data they manage will be stored securely, will be accessible only by people who need it, and so on. In its simplest form, compliance management means confirming and potentially documenting such aspects of the application, and how they are being treated.

Note that most low-code platforms incorporate mechanisms to ensure the application keeps to such rules. However, responsibility falls on the citizen developer to check and assure that this is the case.

Quality management is another area that can start very simple, but that can also benefit from having mechanisms in place upfront, in case things get more complicated later. In its most fundamental form, quality is about making sure that a low-code application works as expected once it reaches its users. For the smallest applications, the citizen developer needs to act as both creator and assessor, confirming all is well prior to distribution.

The question is: how? In general, the answer is to run through everything that the application does, to make sure it works as expected. This is called functional testing. To do this, it is useful to have a checklist of functions, screens, or forms to test, which can then be ticked as they are confirmed. Other kinds of testing, such as performance testing or user experience testing, can follow the same approach if necessary.

A deeper goal of testing is not to prove correctness but to save time. If the citizen developer spots that something is not working correctly before it is passed to someone else, it can be fixed there and then. If it is found by an end-user of the application, the cycle could be much longer and may have knock-on effects, such as reducing productivity or damaging trust.

As we can see, governance areas do not have to be complicated, but it helps to have them in place from the outset, even if a paper-based model is used. Low-code platforms should have capabilities to support the different types of management and may even address them directly—for example, to deal with security and data management, or assist with change and quality management. We cover the expected set of capabilities in our up-coming Key Criteria report on low-code platforms.

5. How to Build a Successful Citizen Development Practice

It isn't difficult to start addressing governance with the right mindset. The approach requires care and consideration in delivery and a recognition that even the simplest of situations, use cases, and applications can become more complex over time. Depending on the situation, we recommend the following:

- **See governance in terms of the minimum necessary.** For example, put in place the elements that you feel are essential, without creating bureaucratic overheads, which would undermine the point of low code.
- **Create a checklist of mandatory principles**, which can be added over time. Keep this as simple as possible; in that way, you'll be more likely to stick to it.
- **See best practices and processes as products**, which can evolve with the application(s) concerned. Version One of your best practice guide will show that you mean to do things right.
- **Incorporate peer review gates**, even if these are with team members. An observer's eye cast over a solution can be an enormous help, and even simple reviews and demos encourage buy-in.
- **Do jump in, but learn as you go**. The best lessons come from trying, as advocated by Fail Fast principles: but these also base themselves on keeping visibility on what works and what doesn't.
- **Low code doesn't mean low skills**. While low-code platforms lower the barrier to entry, citizen developers can learn and build experience over time, both in terms of how to drive the platform and around ancillary areas such as collaborative design.
- **Bring in low-code champions**. Citizen developers can be seen as a symptom of agile maturity as well as an on-ramp to it. Broadcast your successes and encourage more people to think for themselves.
- **Create a management dashboard**. While the simplest of applications may only need a paper- or spreadsheet-based system, more complex applications can benefit from an overview of all management areas.
- **Use low code to manage low code**. Interestingly, low code can be used to build the very management tools and dashboards we talk about here, for overseeing application delivery.
- **Look for adjacencies with best practice groups**. Learn from others using the same tools, about how they are keeping on top of complexity without slowing themselves down; and engage with internal standards teams to follow similar terminology and approaches.

Specific actions will depend on an organization's position on the journey, and the types of applications developed. For example, early on, you can put in place minimum necessary principles and peer review gates. Later on, as the scenario becomes more complex, you may want to lock down your change and quality management approaches.

6. Conclusion – Start on the Right Foot

At the end of the day, citizen development is a term being applied to capture a growing reality in many organizations. It doesn't matter what it is called, but the trend will continue as low-code platforms become more powerful. We are already seeing more features added, such as capabilities to support particular areas of the business, or indeed to deliver on the needs of healthcare, insurance, retail, and other sectors. Organizations need to take the steps they can to harness this reality and make it work for them, rather than see it as something to be avoided or restricted.

To enable this, we advocate a minimum-necessary management approach, with an emphasis on both terms—to do what matters, and not overload with bureaucracy and overheads. Start on the right foot and follow the advice in this guide. Then an organization can move forward with confidence, empower its staff, and enable results without creating problems down the line.

7. Further Reading

In the paper, we have applied a number of areas of thinking to low code and citizen development, showing how some of the best traditions of software best practice also apply to this brave new world. Readers interested in knowing more may be interested in the following books:

- Eliyahu M. Goldratt, Jeff Cox. ***The Goal: A Process of Ongoing Improvement*** (1984). Written as a novel, this book covers process optimization and how it can be applied to project management and innovation.
- Steve McConnell. ***Code Complete*** (1993). This book set out a programming first approach to project management and espoused a minimum necessary approach towards software governance.
- Mike Hammer, James Champy. ***Reengineering the Corporation*** (1993). Business process re-engineering was a massive trend, driving how software could be built to meet the needs of existing and future business activities
- Kent Beck. ***Extreme Programming Explained*** (1999). This book was instrumental in how organizations moved from slow, monolithic approaches to software development, towards methods that put programmers in the driving seat.

8. About Jon Collins

Jon Collins has advised the world's largest technology companies in product and go to market strategy, acted as an agile software consultant to a variety of Enterprise organizations, advised government departments on IT security and network management, led the development of a mobile healthcare app and successfully managed a rapidly expanding Enterprise IT environment. Jon is frequently called upon to offer direct and practical advice to support IT and digital transformation strategy has served on the editorial board for the BearingPoint Institute thought leadership program and is currently a columnist for IDG Connect.

Jon wrote the British Computer Society's handbook for security architects and co-authored The Technology Garden, a book offering CIOs clear advice on the six principles of sustainable IT delivery. He has written innumerable papers and guides about getting the most out of technology and is an accomplished speaker, facilitator, and presenter.

9. About GigaOm

GigaOm provides technical, operational, and business advice for IT's strategic digital enterprise and business initiatives. Enterprise business leaders, CIOs, and technology organizations partner with GigaOm for practical, actionable, strategic, and visionary advice for modernizing and transforming their business. GigaOm's advice empowers enterprises to successfully compete in an increasingly complicated business atmosphere that requires a solid understanding of constantly changing customer demands.

GigaOm works directly with enterprises both inside and outside of the IT organization to apply proven research and methodologies designed to avoid pitfalls and roadblocks while balancing risk and innovation. Research methodologies include but are not limited to adoption and benchmarking surveys, use cases, interviews, ROI/TCO, market landscapes, strategic trends, and technical benchmarks. Our analysts possess 20+ years of experience advising a spectrum of clients from early adopters to mainstream enterprises.

GigaOm's perspective is that of the unbiased enterprise practitioner. Through this perspective, GigaOm connects with engaged and loyal subscribers on a deep and meaningful level.

10. Copyright