13 REASONS CIOs WORRY ABOUT CITIZEN DEVELOPERS BUILDING ENTERPRISE APPS

Krista

VentureBeat

John Michelsen

CEO Krista Software, KristaSoft.com

Businesses need to operate faster and more efficiently to survive. They need more digital capabilities—now. But most enterprise IT organizations have significant supply constraints. There are simply too many business demands and too few skilled developers to deliver new solutions. The number of requirements IT departments receive far exceeds their capacity to fulfill them. The backlog of change requests often number in the hundreds or thousands and are months or years of labor long. Long delays frustrate business leaders and cause them to seek alternative solutions for digital transformation projects. One remedy for this bottleneck that has gotten a lot of attention recently is shifting application development labor from IT to business users. These so-called "citizen developers" create applications for themselves or others, using tools that are not actively forbidden by IT or business units. While it might sound like a great idea, remember the problems brought about by shadow IT, when non-IT workers brought devices, software, and services into their organizations outside the ownership or control of IT. Shadow IT wreaked havoc in organizations when workers installed MS Access on their desktops and created their own databases.

Krista

We can expect to see similar problems as a result of the current citizen developer movement.

Two popular technologies citizen developers use to build new apps are robotic process automation, RPA, and low code application platforms, LCAPs. RPA helps automate tasks typically using UI-based record and playback technology, eliminating the need to integrate systems in a workflow. The user interface is the integration layer so users can bypass system connectivity that requires IT. LCAPs enable business technologists to build apps outside of IT controls. Both tools enable citizen developers to build new apps or hire third-party firms to avoid IT delivery backlogs and delays.

Democratizing technology and enabling non-IT resources to build apps sounds excellent, but this can cause downstream problems for the CIO and enterprise IT. Distributing this work to less trained people makes more work down the road, segregates enterprise data, and introduces more risk to the business because "citizen developers" are not developers.

As long as your citizen developers are not interacting with IT systems or producing data that requires enterprise security and management, your CIO has nothing to worry about. But if that changes, things get complicated very quickly.

Here are 13 reasons a CIO doesn't want citizens developing their own enterprise apps, ordered from the least important to the most important.

13 Apprenticeship is lost

Brand new developers joining IT don't start by creating mission-critical apps without oversight. Instead, they are mentored by senior developers who have both formal and informal education about what works and what fails in their enterprise. With a citizen development team this guidance is lost, and the risk of costly development errors is high.

12 Deploying and managing platforms is no different

As soon as the app in question is accessing mission-critical or sensitive data, IT must extend its change management processes to that platform. That means dev environments, test environments, integration environments, performance test environments, and others. We hold IT accountable for system and data integrity; thus, these steps are necessary. Your citizen developers will essentially build apps under the same processes that IT follows.
These apps are thus subject to the same development delays as your IT-built apps. Most delays are due to environment and test data availability and management. If this development aspect is the same, citizen development won't be any faster than traditional IT development.

11 Separation of duties

In software development, there is a firm separation of duties. Strict governance doesn't allow developers to perform their own quality assurance, so errors are caught before production (hopefully!). After a few unexpected "sev 1" issues, the citizen development process will be forced to mirror that of existing IT development practices to ensure requirements are properly captured, code is tested by independent quality assurance people, and changes are deployed cautiously.

10 Economics

Building RPA apps to automate repetitive processes may seem like a cost saver. However, most of the people building these apps for the business are with third-party service firms. For instance, companies spend four dollars on services for every dollar spent on RPA software licenses. Spending so much on services to create or edit automations increases the total cost of ownership and may not have been accounted for at the beginning of the project. And since IT also leverage third parties for much of its software development, there's again no good argument here for bypassing IT in the first place.

9 Security posture

Citizen developers are everyday employees who introduce security risks to the business. They often employ poor security practices like reusing passwords, leaking data, and not keeping systems up to date. Therefore, companies can expect to spend billions of dollars on security software like firewall protection, antivirus, and anti-phishing software to protect the organization and lower the risk of inadequate security practices and hygiene from "citizens." The Infosec team's governance of IT software projects must extend to these projects no differently.

8 Control and governance

IT governance combines rules, regulations, and policies that define, control, and ensure effective operations of an IT department. Democratizing technology and allowing non-IT employees to build applications can cause data and processes that weaken governance and centralized ROI reporting. This is especially true if data created inside a citizen-built app isn't available for enterprise reports and dashboards. The absence of proper governance of citizen development projects either limits their scope dramatically or represents dangerous activities that must be brought under the same control structure as other IT initiatives.

7 Citizens don't want to do it

So-called "citizens" aren't necessarily excited by being given such "power" to develop apps. It's not a matter of tools and technology but whether or not they were hired to perform such tasks. There is always a fraction of non-IT people interested in app development; those people typically make their way into IT roles. Those who are not interested want to use technology, not create it.

6 Task orientation – the opposite of the big picture

Typically, citizen developers only partially automate the steps they take in a process, not the end-to-end business process outcome.
Without a big picture view, we fall victim to Constrain Theory and end up with suboptimizations that may or may not produce an actual ROI on the outcome desired.

5 Makes transformation harder

This task orientation of low-code platforms will often make business transformation harder. These platforms expose the business logic they embody via a UI. They are built for people to launch and click around. So building automation end-to-end, i.e., incorporating that logic in a larger context, becomes an even more challenging proposition than before the app was created.

Hardcoding the way tasks are performed today is often not getting you closer to a transformed outcome. How do we take six steps down to 2, or even 1? That's not the goal of most low-code platforms, and it's not a goal that citizen developers have in scope.

4 Production outages are difficult to triage

Enterprise applications with various people and system integrations get pretty complex. Understanding issues and resolving them often takes experts representing the many systems involved. Folks in IT know all too well about the 50-person conference call to triage a high severity issue. IT must run production support and have significant involvement if a system is to remain up. Otherwise, downtime could ruin the value of the whole low-code initiative.

3 Most low code tools oversell capabilities

Many LCAPs state that developing apps using their platforms is easy and fit the citizen developer model, but "low code" doesn't mean "no code." When it comes to integrating with other systems, they follow what we call the "paste your code here" model. One LCAP developer stated on Gartner Peer Insights, "Processes that require business logic beyond what is built and available off the shelf require professional developers. I personally have worked to develop over 50 applications and would not have been able to develop a single one without the support of professional developers."

It takes an average of 101 days of training, mentoring, or upskilling citizen developers to overcome the skills gap problem. Just go to your favorite job posting board and look at the requirements of low/no-code platform jobs. You'll see they require five years of Java and three years of SQL experience! What's the difference between those postings and typical IT developer postings?

2 Businesses already have too many apps

This one really gets me going. Here we are as an industry trying very hard to make new web app building faster and faster. But what business leader ever said, "What my team needs are more apps to deal with!"

Businesses are already overwhelmed with the burgeoning list of apps in the workplace. An enterprise uses 397 apps on average. These apps have separate user interfaces and terminology, functional features, license costs, and/or a development team with a backlog of change and support requests. The average employee trying to manage processes through all of these apps switches between 35 job-critical applications more than 1,100 times every day. More apps increase costs and frustrate employees.

1 A productive "citizen developer" is a "developer"

Note how many of the concerns above essentially resolve to "citizen developers have to do the same thing IT already does"? If they are doing all the same things a developer in IT has to do, they too are developers. By the time you get the citizen developer productive and safely contributing, you might as well drop the word "citizen."

This article originally appeared as a guest post on VentureBeat.com.

About the Author

John Michelsen, CEO Krista Software,
KristaSoft.com

John has invested himself in pushing the leading edge of technology transformation to deliver business outcomes and is a highly respected technologist who moves others to action.

John is a proven technology leader with 28 patents awarded or in process in database, distributed computing, virtual/cloud management, multi-channel web application portals, Service Virtualization (LISA), and the industry's most advanced mobile security.

Today, John is CEO of Krista Software and has conceived a whole new way to automate business processes and deliver digital capabilities.