

Citizen Developers from City Dwellers to Townsmen



Evolving Citizen Developers from City Dwellers to Townsmen ... so they can become first class citizen

Citizen developers have triggered **a new left-shift in the application development.**

With the introduction of DevXOps, in its multitude of variations DevSecOps, DevBizOps, DevDataOps ... we have seen **a first shift toward putting more responsibilities with the developers of applications.** The idea that, if we **consider** security, usability, manageability, operability, monitorability, business value and other ***-abilities as quality attributes from the get-go** of the development process and not wait till applications get into production, resulted in new application life cycle management (ALM) processes. ALM processes that focus on continuous testing, continuous integration, continuous deployment, continuous feedback, continuous monitoring and continuous operations (**7 C's op DevOps**). New ALM processes that support this left-shift from operations to development and can even exonerate developers from these additional responsibilities.

Low-code platforms targeted initially developers with the promise of acceleration introducing the concept of Rapid Application Development (RAD). Using drag and drop features to create UI's; visual wizards to add business logic for validation and other business rules into applications; and schema generation to expose data

to applications all resulted in faster time-to-market of IT solutions.

After the introduction of low-code (L-C) platforms, **a new group developers** raised to the occasion. **Tech savvy business users** started to use the same L-C platforms to solve day-to-day problems that could not be picked up quickly by IT departments through traditional or rapid application development. **The citizen developer was born**. At the beginning this was done covert and resulted in **shadow-it**. When L-C platforms evolved towards no-code (N-C) capabilities, the need for **governing this new group** of developers became apparent. This **resulted in a second left-shift, now from pro-developers to citizen developers**. Also here, responsibilities shifted and this time across the IT – Business chasm.

What are these responsibilities that landed in the laps of the Citizen Developers?

Typically we see first the same responsibilities be shifted, that were apparent already in the first left-shift, from operation to pro-developers. These are mostly **IT related risks**:

-  **ALM risks: keeping the application up to date** through evolving the requirements, architecture and technology of the application **to protect against functional obsolescence and technical debt.**
-  **Security Risks: keeping the application secure** and **protect against incorrect access, usage and data exposure.**
-  **Costs Risks: baring the cost of the application** and **protect against inefficient and uncontrolled spending**
-  **Governance Risks: ensure the application is managed as part of a solution architecture landscape**

Typically when Citizen Development matures, we see that also **business related risk** become visible. These should be part of the responsibility shift as well. These Business related risks are:



- **Strategic Risks:** Citizen developer's focus limited to **ad-hoc problem solving** i.e. operational problems, **forgetting about tactical and strategic** long-term **solutions**. Missed opportunities towards generalization and standardization of solutions leading to redundancy/duplication: **excess costs and limited scaling**.



- **Process Risks:** Citizen developer's focus on a **part of a business process** (one or more activities) resulting in **sub-optimal solutions**. Automating an inefficient process or part of a process results in an automated inefficiency. It requires an end-to-end process view and level of abstraction beyond the problem at hand to create efficient solutions.



- **Obsolescence Risks:** Citizen developers **applying a technology because they can, not because it is the right thing to do**: *"If you have a hammer everything looks like a nail"*. Limited critical reflection on why and when to apply a technology. **Lack of understanding** of correct usage and limitation of a technology.



- **Regulatory Risks:** Citizen developers' focus solely on the business process **forgetting about regulatory compliance** e.g. data usage (GDPR, data privacy) and security (data leakage and data loss). **Limited understanding of requirements** beyond functional requirements omitting domain and non-functional requirements. **Limited accountability toward end-to-end process and compliance of activities** part of an end-to-end process.

Most of these risks are managed from within a modern L-C /N-C platform through tooling and monitoring. These are basically **where the platforms turn shadow-it into business managed solutions**. The business risks are often underexposed in these platforms and **managed by means of on-boarding checklists, ideation repositories and usage monitoring**.

The latter typically requires a strategy and governance added on top what these platforms can offer. The platform is a means to the end but cannot exonerate an organization of all the responsibilities that were shifted to the Citizen Developers.

*It is the tooling support of the L-C/N-C platforms
that evolve Citizen Developers from City Dwellers to Townsmen.*

It is the organization's (city) strategy and tactical focus that turns them into first class citizens.

The strategy and tactics will ensure that the minimal conditions are created for Citizen Development success. Typically we see 3 prerequisites for the success, the so-called **D³ i.e. Data, Devices and Delivery**

Data: data encapsulation/exposure is a prerequisite



For a Citizen Developer to develop applications for his business, he needs access to his business data

- **Data source isolation:** data specific for his business
- **Data availability:** data accessible through API's
- **Data storage:** data to be stored in relation to his application

Devices: suitable devices are a prerequisite



For a Citizen Developer to build applications, he needs to be able to select a suitable device for the problem at hand i.e. desktop, mobile, VR-AR, kiosk

- **Device availability:** devices made available on short-term to all the users of his application
- **Device/License sharing:** devices and licenses ad-hoc attributed and revoked to users
- **Device security policies:** devices are managed to protect against exposure and loss of business data

Delivery (Deployment): platform governance and strategy is a prerequisite



*For a Citizen Developer to manage the life cycle of his application,
he needs to be able to count on supporting process to be in place*

- **Application Delivery TOM/SOM:** application delivery processes clearly describe roles and responsibilities for Business and IT
- **Delivered Application Support Model:** delivered application he can support directly or indirectly
- **Application DevSecOps policies:** business risks and IT risks are controlled by DevSecOps tooling