# Technical Debt in Low-code platforms

**Cecilia Apa**

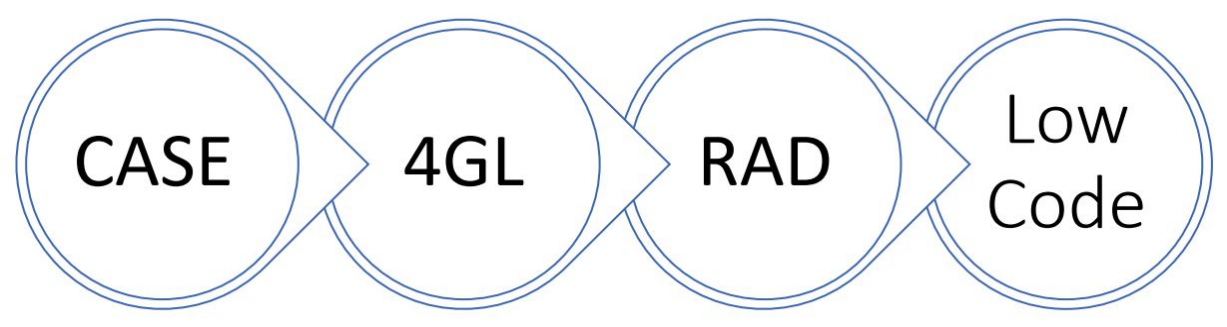
ceapa@fing.edu.uy

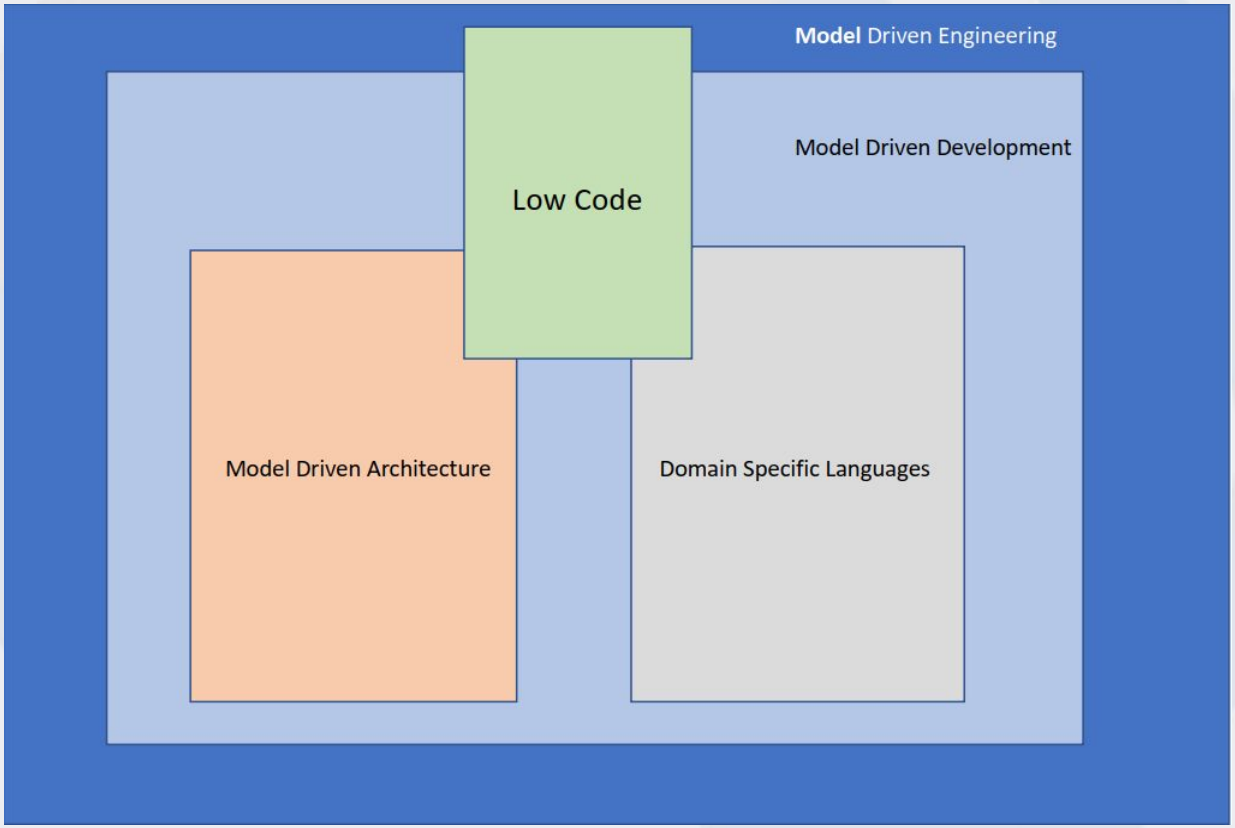
# Low-code development platforms (Specific development context)

- **Low-code** - introduced by the industry analyst Forrester Research (2014)

Wikipedia definition

*A low-code development platform (LCDP) is software that provides a development environment used to create application software through graphical user interfaces and configuration instead of traditional hand-coded computer programming. A low-code model enables developers of varied experience levels to create applications using a visual user interface in combination with model-driven logic.*




"Plataformas Low-Code - ¿Qué hay de nuevo viejo? JIS.uy 2019 talk

# LCDP Benefits

- Reduce the amount of traditional *hand coding*
- Enabling accelerated delivery of business applications.
- Lower the initial cost of setup, training, deployment and maintenance.
- A wider range of people can contribute to the application's development (not only those with formal programming skills).

# Technical Debt and LCDP

- There is incipient scientific literature on Technical Debt and LCDP

No documents were found.
Show results for: TITLE-ABS-KEY ( "TECHNICAL DEBT" AND ( "LOW-CODE" OR "LOW CODE" OR "cdp" ) )

- In some grey literature LCDP has been “promoted” as a solution to certain TD problems.

# LCDP grey literature

- "Avoiding the Legacy Trap With Low-Code" [1]
  - "When developers can build features and applications faster, they have more time to pay down technical debt"
  - "Avoiding coding-related technical debt is easy with OutSystems because best practices are built into the platform."
- "Benefits of Low-Code" [2]
  - "Low-code makes you forget about app obsolescence and technical debt with built-in safeguards to ease application changes and maintenance"
- "How to Transform Technical Debt into Toast" [3]
  - "One of the most significant sources of technical debt: hand-coding."
  - "Low-code development platforms, on the other hand, offer organizations a way to reduce the accruing of technical debt."

# LCDP grey literature

- “TECHNICAL DEBT - THE PROMISE AND PERIL OF LOW-CODE APPLICATIONS” [4]
  - “While powerful, enterprise systems can be technical debt traps.”
  - “If business users are empowered to build their own tools and can build them rapidly, we are trading one form of technical debt for another.”
- “What CIOs need to know about low code software development” [5]
  - “Low code without integration, for this reason, is viewed as a tech debt expander.”

# What the Uruguayan software practitioners said?

- In the MIS.uy Focus Group
  - They experienced TD in a particular Low-Code platform (GeneXus)
  - There is no software tools that helps to identify, measure and monitor TD in LCDP.
- I was a LCDP practitioner (many, many years ago)
  - I experienced the concept of TD in the software that I developed and maintain at this time.

# Motivation

- There is no scientific research about Technical Debt in the context of LCDP
- There is a concern from software industry practitioners on how to manage TD using LCDP
  - Lack of tools
- LCDP has been promoted as a solution to some TD problems

# Research method

- Select a representative LCDP to investigate TD
  - To Characterize TD
  - Identify similarities and differences between No-LCDP

# GeneXus

- Uruguayan software company founded in 1988
  - GeneXus™ 1.0 was released in 1989
  - Today:

# GeneXus in Uruguay

- Strong community
  - GeneXus meetings (Uruguay, Chile, Colombia, Japan, China)
    - Gx29 – 4000 participants from 30 different countries.
- Used in large companies and in government agencies
- Evaluated by Forrester as "the best low-code platform you've never heard of".
- There is a need of improve TDM from the Uruguayan practitioners that use GeneXus.

# References

[1] Avgeriou, P., Kruchten, P., Ozkaya, I., Seaman, C.: Managing Technical Debt in Software Engineering. Dagstuhl Reports 16162, Germany (2016)

[2] https://www.outsystems.com/blog/posts/avoiding-legacy-trap-low-code/

[3] https://lowcode.com/what-is-low-code/

[4] https://www.outsystems.com/blog/posts/transform-technical-debt/

[5] https://www.meraksystems.com/blog/2019/10/14/technical-debt-the-promise-and-peril-of-low-code-applications.html

[6] https://www.cio.com/article/3410878/what-cios-need-to-know-about-low-code-software-development.html