Positioning of the low-code movement within the field of model-driven engineering

Jordi Cabot
ICREA
UOC
Spain
jordi.cabot@icrea.cat

ABSTRACT

Low-code is being promoted as the key infrastructure for the digital transformation of our society. But is there something fundamentally new behind the low-code movement? How does it relate to other concepts like Model-Driven Engineering or Model-Driven development? And what are the implications for researchers in the modeling community?. This position paper tries to shed some light on these issues.

CCS CONCEPTS

• **Software and its engineering → Software design engineering**; **Software prototyping**; *Unified Modeling Language (UML)*; *Specification languages.*

KEYWORDS

Software Modeling,low-code, no-code, model-driven

ACM Reference Format:
Jordi Cabot. 2020. Positioning of the low-code movement within the field of model-driven engineering. In *ACM/IEEE 23rd International Conference on Model Driven Engineering Languages and Systems (MODELS '20 Companion), October 18–23, 2020, Virtual Event, Canada.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3417990.3420210

1 INTRODUCTION

Low-code application platforms accelerate app delivery by dramatically reducing the amount of hand-coding required [1]

This is clearly not the first time the software engineering community attempts to reduce manual coding by combining visual development techniques (what we would call "models") and code generation. In fact, as Grady Booch says, the entire history of software engineering is about raising the level of abstraction. **Low-code can be traced back to the model-driven engineering**. But model-driven engineering itself can be traced back to CASE (Computer-Aided Software Engineering) tools. Already in 1991, in the 1st edition of the well-known CAiSE conference, we could find papers stating concepts like: *"Given the final model, the complete computerized information system can be automatically generated"*[3] or *"we arrive at a specification from which executable code can be automatically generated"*[5].

At the same time, the impact of low-code in the business world is also evident nowadays, including some bold projections[2] but also actual factual numbers regarding recent investments in low-code tools, the commercial success of some of them or just the fact that all the largest software companies are making sure they have some kind of offering in this domain [3].

The rest of the paper will discuss in more detail these two, apparently contradictory, statements. Next section aims to clarify the relationship between low-code and other related modeling terms. Section 3 looks for the reasons behind the low-code popularity. Finally, Section 4 wraps up the paper by reflecting on whether the low-code movement is in the end good or bad for the modeling community.

2 LOW-CODE VS MODEL-DRIVEN VS MODEL-BASED VS NO-CODE

We do not have universal definitions for all the model-*. The MBE-BOOK [2] does a great job in clarifying specific terms but does not cover more methodological aspects, moreover low-code/no-code concepts are not part of it). Therefore, my own (informal) definitions are the following:

- Model-driven Engineering (MDE): any software engineering process where models have a fundamental role and drive the engineering tasks.
- Model-driven development (MDD): MDE applied to forward engineering, i.e. model-driven for software development.
- MDA is the OMG's particular vision of MDD and thus relies on the use of OMG standards.
- Model-based engineering/development: Softer version of the previous respective concepts. In a MBE process, software models play an important role although they are not necessarily the key artifacts of the engineering/development (i.e. they do NOT "drive" the process).

An example of the MBE vs MDE difference would be a development process where, in the analysis phase, designers specify the platform-independent models of the system but then these models are directly handed out to the programmers to manually write

[1] Definition taken from this Forrester report [6], attributed as the origin of the term *low-code.*

[2] A couple of examples. According to Gartner, by 2024, low-code application development will be responsible for more than 65% of application development activity. And Forrester expects the low-code market to represent $21B in spending by 2022.
[3] https://modeling-languages.com/big-five-bet-modeling-low-code/

MODELS '20 Companion, October 18–23, 2020, Virtual Event, Canada

ACM ISBN 978-1-4503-8135-2/20/10…$15.00
https://doi.org/10.1145/3417990.3420210

the code (no automatic code-generation involved and no explicit definition of any platform-specific model). In this process, models still play an important role but are not the basis of the development process.

Based on the above definitions, I see **low-code as a synonym of model-driven development**. If anything, we could see low-code as a more restrictive view of MDD where we target only a concrete type of software applications: data-intensive web/mobile apps. As suggested in an online discussion around these topics[4], low-code can also be regarded as a fixed-language solution as the language itself behind the low-code tool is typically not exposed and cannot be changed, while in a MDD solution you do have the flexibility to define/adapt the language to be used.

Note that the term no-code is sometimes used as a slight variation of low-code. In fact, we can often see tools defining themselves as *no-code/low-code* tools. Nevertheless, to me, the key characteristic of a no-code approach is that app designers should write zero code to create and deploy the application. This limits a lot what you can actually do with no-code tools. We are basically looking at template-based frameworks or creation of workflows mixing predefined connectors to external applications where the designers, at most, decide when and how certain actions should be triggered.

Another way to compare these different paradigms is by looking at how much manual code you are expected to write. In MBE, you may have to write all the code. Instead, in MDD and low-code, most of the code should be generated but you still may need to customize and complete the generated code[5]. In no-code you should write zero code.

Obviously, more research is needed to evaluate the low-code tools in the market and better characterize them in less coarse-grained categories than those presented here. In fact, right now, there is basically no research around the low-code movement[6], something that I am sure this workshop will start to change.

3 LOW-CODE IS TRENDING

As shown in the Figure 1, interest in low-code is as its peak, even if, as depicted in Figure 2, this peak is much smaller than the attention model-driven was getting on its prime.

But, if, technically speaking, low-code does not really bring anything new to the table, why this popularity?.

First of all, I think low-code conveys a much clearer message than model-driven/model-based. "Model" is a much ambiguous word and therefore the concept of model-driven is more difficult to explain than low-code (everybody has a clear view of what code is, and low-code becomes self-explanatory).

Secondly, we know modeling scares developers away. Instead, low-code sounds more familiar. It is the same they already do (coding) but less of it.

Moreover, the application scenarios for low-code are also clearer. Instead of selling that you can do anything with MDD (which ends up generating mistrust), low-code looks more credible by targeting specific types of applications, those that are most needed in industry.

Low-code is also typically a one-shot modeling approach, meaning that you have models and the generated code, no complex chains of refinement, no model transformations, no nothing.

And on average, low-code tools are nicer than our traditional heavy modeling tools. For instance, most are web-based and do not depend on EMF.

All in all, I haven't seen any notation, concept, model type or generation technique in a low-code tool that I couldn't find similarly in the model-driven world. But for sure, these same techniques are presented, configured, adapted and "sold" differently, which in the end makes a big difference in how low-code novelty and usefulness are perceived. And the success of a MDE project often depends more on social and managerial aspects than on purely technical ones [4]. This does not come for free (lack of interoperability, vendor lock-in, expensive business models,..) but this does not seem to deter the community at the moment.

4 LOW-CODE AS AN OPPORTUNITY

As pointed out before, I do not believe there is any fundamental technical contribution in low-code trend. It is more a synonym (or a subset) of the techniques we are already working on. In fact, we could take almost any of the open challenges in model-driven engineering [1] and just change "model-driven" by "low-code" to get, for free, a research roadmap for low-code development (e.g. we need better ways to integrate AI in low-code tools or we should strive as a community to build a shared repository of low-code examples for future research).

But I do not see this as being negative. More the opposite. Clearly, low-code is attracting lots of attention, including from people that were never part of the modeling world. In this sense, low-code is lowering the barrier to enter the modeling technical space. As such, to me, **low-code is a huge opportunity** to bring modeling (and our modeling expertise) to new domains and communities. If we can get more funding/exposure/users/feedback by rebranding ourselves as low-code experts, I am all for it. This is exactly the approach that many well-known so-called low-code companies have taken[7]. Let's also take this opportunity to better understand the factors that make modeling-like techniques resonate in the broad software community and learn from it. The lack of a strong open source community around the low-code movement could also be a opportunity for us as we have plenty of experience in building opens source modeling tools and discussing the trade-offs of doing so.

And while we do that, let's keep an eye on the market trends to come. Some low-code vendors are shifting (yet again) their marketing efforts. It may not be long before we all start chanting: *Low-code is dead, long live multi-experience development*[8]

Acknowledgements. Thanks to Manuel Wimmer for our very interesting discussions on this topic.

[4] https://modeling-languages.com/low-code-vs-model-driven/

[5] Most MDD tools include some kind of black box modeling primitive where you can write any custom code that should be added during the generation process

[6] A quick search only reveals some papers about tools that classify themselves as low-code but not about low-code itself as the object of study

[7] Feel free to play with the Internet Wayback Machine and see how their websites mutate from visual modeling, agile development, CASE tools and similar keywords to low-code in the last years

[8] Multiexperience refers to the various permutations of interactive modalities (e.g., touch, voice and gesture) apps must offer now https://www.gartner.com/en/information-technology/glossary/multiexperience-development-platforms-mxdp

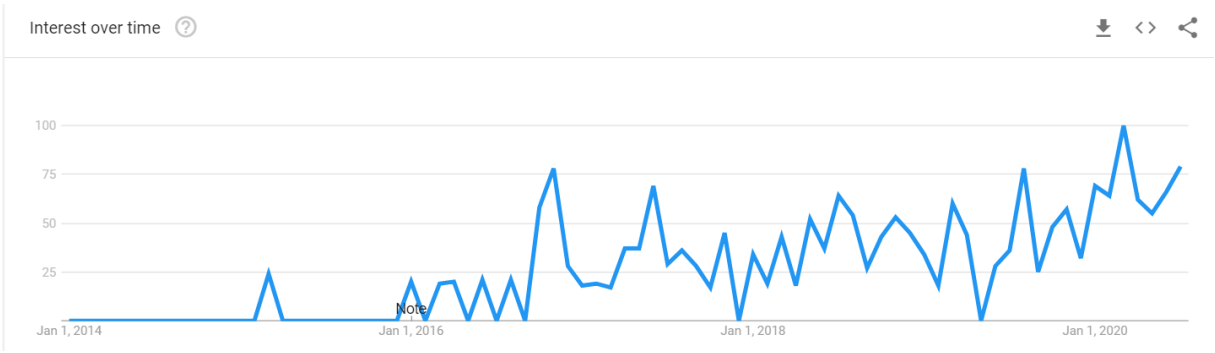

Figure 1: Google Trends graphic showing the search interest for the low-code term

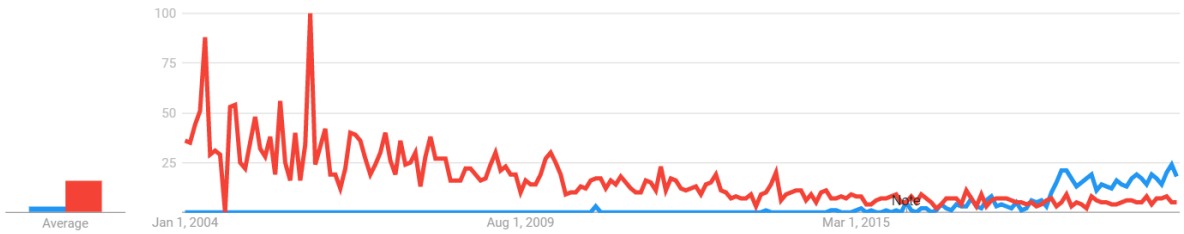

Figure 2: Google Trends graphic showing the relative search popularity of model-driven (red) vs low-code (blue). Interest over time represents search interest relative to the highest point on the chart to get a sense of the relative search size of both terms.

REFERENCES

[1] Antonio Bucchiarone, Jordi Cabot, Richard F. Paige, and Alfonso Pierantonio. 2020. Grand challenges in model-driven engineering: an analysis of the state of the research. *Software and Systems Modeling* 19, 1 (2020), 5–13. https://doi.org/10.1007/s10270-019-00773-6

[2] Loli Burgueño, Federico Ciccozzi, Michalis Famelis, Gerti Kappel, Leen Lambers, Sébastien Mosser, Richard F. Paige, Alfonso Pierantonio, Arend Rensink, Rick Salay, Gabriele Taentzer, Antonio Vallecillo, and Manuel Wimmer. 2019. Contents for a Model-Based Software Engineering Body of Knowledge. *Software and Systems Modeling* 18, 6 (2019), 3193–3205.

[3] Jon Atle Gulla, Odd Ivar Lindland, and Geir Willumsen. 1991. PPP: A Integrated CASE Environment. In *Advanced Information Systems Engineering, CAiSE'91, Trondheim, Norway, May 13-15, 1991, Proceedings (Lecture Notes in Computer Science, Vol. 498)*. Springer, 194–221. https://doi.org/10.1007/3-540-54059-8_86

[4] John Edward Hutchinson, Jon Whittle, and Mark Rouncefield. 2014. Model-driven engineering practices in industry: Social, organizational and managerial factors that lead to success or failure. *Sci. Comput. Program.* 89 (2014), 144–161. https://doi.org/10.1016/j.scico.2013.03.017

[5] John Krogstie, Peter McBrien, Richard Owens, and Anne Helga Seltveit. 1991. Information Systems Development Using a Combination of Process and Rule Based Approaches. In *Advanced Information Systems Engineering, CAiSE'91, Trondheim, Norway, May 13-15, 1991, Proceedings (Lecture Notes in Computer Science, Vol. 498)*. Springer, 319–335. https://doi.org/10.1007/3-540-54059-8_92

[6] Clay Richardson and John R Rymer. 2014. New Development Platforms Emerge For Customer-Facing Applications. *Forrester: Cambridge, MA, USA* (2014).