Lowcomote: Training the Next Generation of Experts in Scalable Low-Code Engineering Platforms

Massimo Tisi[1], Jean-Marie Mottu[1], Dimitrios S. Kolovos[2], Juan de Lara[3], Esther Guerra[3], Davide Di Ruscio[4], Alfonso Pierantonio[4], and Manuel Wimmer[5]

[1] LS2N, IMT Atlantique, Université de Nantes, France
[2] University of York Helsington, UK
[3] Universidad Autónoma de Madrid, Spain
[4] Università degli studi dell'Aquila, Italy
[5] JKU Linz, Austria

{massimo.tisi, jean-marie.mottu}@imt-atlantique.fr,
dimitris.kolovos@york.ac.uk, {juan.delara, esther.guerra}@uam.es,
{davide.diruscio,alfonso.pierantoni}@univaq.it, manuel.wimmer@jku.at

Abstract. Low-Code Development Platforms (LCDPs) are software development platforms on the Cloud, provided through a Platform-as-a-Service model, which allow users to build completely operational applications by interacting through dynamic graphical user interfaces, visual diagrams and declarative languages.
Lowcomote will train a generation of experts that will upgrade the current trend of LCDP to a new paradigm, Low-Code Engineering Platform. This will be achieved by injecting in LCDPs the theoretical and technical framework defined by recent research in Model Driven Engineering, augmented with Cloud Computing and Machine Learning techniques.

1 Project Identity

- **Project acronym:** Lowcomote
- **Project title:** Training the Next Generation of Experts in Scalable Low-Code Engineering Platforms
- **Partners:** Institut Mines Telecom (Coordinator), University of York, Universidad Autónoma de Madrid, University of L'Aquila, Johannes Kepler Universität Linz, British Telecom, Intecs Solutions, Uground, CLMS, IncQuery Labs, The Open Group, Metadev, SparxSystems, Amazon Web Services
- **Website:** `www.lowcomote.eu`
- **Funding:** EU H2020-MSCA-ITN-2018 - Innovative Training Networks (ITN)
- **Project start date/duration:** 1st January, 2019 (48 months)

2 Introduction

A large number of research efforts in the history of computer science have shared a common objective: enabling the construction of software applications without recurring to traditional procedural computer programming. Early products of these research efforts, like fourth-generation programming languages and rapid application development tools, have seen a significant industrial adoption in alternating periods since the nineties, but never reached a dominating position in the software construction landscape. Today a new generation of tools is successfully addressing this challenge in specific domains. They are commonly called **Low-Code Development Platforms** (**LCDPs**): software development platforms on the Cloud, provided through a Platform-as-a-Service (PaaS) model, that allow users to build fully operational applications by interacting through dynamic graphical user interfaces, visual diagrams and declarative languages. In a recent report [1], the research firm Forrester is forecasting a market increase for LCDP companies to over $21 billion over the next four years. Major PaaS players are all integrating LCDPs (e.g., Google App Maker and Microsoft PowerApps [2]) in their general-purpose solutions.

LCDPs build on top of recent advancements in visual programming, automatic code generation, and Cloud infrastructures. LCDPs provide the user with a completely managed environment for the whole application life-cycle, including development, deployment, execution and monitoring. Therefore, they are highly usable by customers with no particular background in programming, called **citizen developers** in the LCDP jargon, without sacrificing the productivity of professional developers. So far LCDPs have been especially successful for the development of domain-specific applications in four market segments, identified in [3]: database applications, mobile applications, process applications, and request-handling applications. Internet of Things (IoT) will be the fifth one.

Lowcomote considers three main limitations that hamper the use of LCDPs. **Scalability**: as LCDPs are currently popular for the development of small applications but their use in large-scale and mission-critical enterprise applications is requested for next evolution. **Fragmentation**: Each tool vendor proposes his own low-code development paradigm, associated with a particular programming model. **Software-only** systems: while citizen developers have little knowledge of programming, they are often experts in some other engineering domain. These domain experts expect to be able to use their knowledge in the application, at the right level of abstraction and using familiar formalisms.

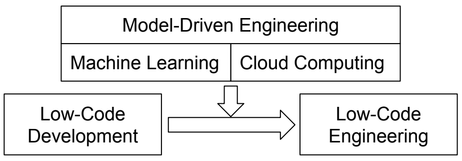

Fig. 1: Lowcomote in a nutshell

Lowcomote is an Innovative Training Network (ITN) aiming to train a generation of professionals in the design, development and operation of new LCDPs, overcoming the current limitations, by being **scalable** (i.e., supporting the development of large-scale applications, and using artefacts coming from a large number of users), **open** (i.e., based on interoperable and exchangeable programming models and standards), and **heterogeneous** (i.e., able to integrate with models coming from different engineering disciplines). They will drive the upgrade of the current landscape of LCDPs to **Low-Code Engineering Platforms** (LCEPs).

The 15 doctoral subjects (`https://www.lowcomote.eu/call/`) in Lowcomote address the hard research problems that are blocking factors for the achievement of these objectives. One of the key research ideas of Lowcomote is that platforms for Low-Code Engineering (LCE) can be built by injecting in LCDPs the theoretical and technical framework defined by recent research in **Model-Driven Engineering** (MDE). MDE research promotes a linguistic approach to software engineering, by the design of domain-specific modelling languages and their automated transformation in executable artefacts. Research in MDE has produced modelling frameworks supported by **open standards**, with a **natural support for heterogeneity**. While scalability has been traditionally problematic in MDE, some Lowcomote partners have been involved in the EC FP7 research project MONDO (2013-2016), which made **substantial breakthroughs on scalable modelling** and model management. Lowcomote builds on top of MONDO theoretical and technical solutions, but upgrades them to enable their use for LCEPs. This is achieved by a multi-disciplinary research program integrating research on **Cloud Computing and Machine Learning**.

3 Main challenges and outlines

This section identifies the main challenges that need to be overcome, and outlines the envisioned contributions of Lowcomote.

3.1 Low-code Engineering of Large-Scale Heterogeneous Systems

LCDPs are typically targeted to users with low technical profile. The supported applications are frequently form-based enterprise systems (sometimes deployed on mobile phones), which are described graphically. However, the approach fails to scale for more complex software – beyond simple CRUD applications – and is not applicable to domains with high impact nowadays, like data science or IoT. Shortening the development times in those domains, by being able to apply a low-code development approach, would be have a considerable economic impact in today's industry.

To scale LCDP for more complex applications and challenging domains, Lowcomote proposes employing solid language engineering principles [4] to generate a new generation of scalable Cloud-based low-code editors. By using abstraction and graph summarisation techniques, such editors will be able to scale to models exceeding millions of elements. At the same time, the platform will be neutral on the underlying modelling technology to avoid vendor lock-in issues, and support heterogeneity (required by e.g., the IoT domain) through multi-view modelling. To help users with low technical profile to develop complex applications, we propose enriching LCDPs with recommendation chatbots. These will interact using natural language and use information retrieval and machine learning techniques to recommend appropriate task completions, or propose example fragments and templates.

Based on the notion of Active DSL [5], LCEP will support citizen development of sophisticated mobile apps, beyond the capacities of current LCDPs. Our targeted apps will be collaborative, and offer interaction based on graphical diagramming, where elements may be geolocated on maps. They will be able to incorporate information from open APIs to access services (e.g., weather), and be context sensitive, able to adapt to changing conditions like device position, time or other conditions retrieved from APIs.

Finally, LCEP will be applied to construct domain-specific LCDPs for data science and IoT. Many tasks of data science projects are currently performed in an ad-hoc way, and solutions many times need to be re-implemented with more scalable and robust technologies before entering in production. Our envisioned LCDP will provide the necessary languages for a high level description of the solution, supporting integrated deployment automation, and scalable execution and monitoring. Similarly, our envisioned IoT platform will enable the scalable modelling of heterogeneous, dynamic data sources and devices, while retaining the low code benefits of reduced development times.

3.2 Large-scale Repository and Services for Low-Code Engineering

Over the last decade, several technologies have been proposed for supporting a wide range of model management activities, such as model validation, transformation and code generation. Even though existing technologies provide practitioners with facilities that can simplify and automate many steps of MDE processes, empirical studies show that some barriers still exist for their wider adoption [6]. In particular, the support for discovery and reuse of existing modelling artefacts is very limited. As a result, similar transformations and other model management tools often need to be developed from scratch, thus raising the upfront investment and compromising the productivity benefits of model-based processes. Lowcomote will advance the state of the art by developing an extensible model repository specifically defined to address issues related to scalability and heterogeneity of the considered modelling artefacts. This model repository will be used to enable capability discovery and reuse across different low-code systems using reinforcement learning algorithms.

Another important aspect of model repositories is which kind of information may be stored and which kind of tools, systems, and applications may contribute content [7,8]. Currently, model repositories are limited for mostly hosting different versions of design models created by development environments [7]. But further support is needed, especially for runtime aspects and runtime information may be manifold [9]. For instance, runtime information may concern the interactions with the development environment and the associated repository which will be explored as interaction mining. Another important kind of runtime information which should be accessible in model repositories is operational data monitored in running systems which may lead to more reactive models usable not only for design but also for supervisory control and data acquisition.

Finally the repository will include a quality workbench (QW). The latter requires to distribute the tests (which are low-code artefacts) and to query the repository to get and to store operational data of the tests, considering scalability [3]. Moreover, the QW will manage the test automation [10] by implementing low-code tests based on existing but distributed low-code artefacts and testing the distribution over the Cloud itself [11].

3.3 Scalable Low-Code Artefact Management

Management and evolution of large-scale LCEPs presents complexity and scalability challenges. Even though MDE provides a number of tools for artefact management, current MDE frameworks fall short when it comes to manipulating and analysing models of the required complexity and scale, both in terms of performance and in terms of resource requirements (e.g., memory footprint) [12,13]. Moreover, currently-available tools for developing and executing model transformations struggle with very large and distributed models over a parallel and distributed environment [14]. The challenge is multiplied in the case of live model transformations, which are continuously run in the background and react to changes and events in the environment [15]. This requires investigating an efficient search-based pattern matching algorithm that can apply to models of millions of elements, and the solution must efficiently run on distributed and parallel environments.

Lowcomote will go beyond the state of the art by developing the theoretical underpinnings and technical components for a model management engine that is able to execute transformations over a highly distributed computing infrastructure, providing scalability both in terms of model size and transformation complexity, and reacting immediately to changes in the environment. The engine will support multiple distributed programming models, automatically selecting the most convenient one, for the whole transformation or only part of it. The engine will support also chains of transformations, with mechanisms for the automatic selection and composition of model transformations in the Cloud.

Lowcomote will advance the state of the art in the field of efficient persistence and querying of large-scale models. Novel techniques and algorithms will be developed to optimise computationally-expensive queries operating on models specified with different modelling languages and in model representation formats. Advanced mechanisms for selective/partial loading and persistence of large-scale models will be proposed and implemented.

4 Conclusion

In this paper we have provided an outline of the main challenges of Low-Code Engineering and how the Lowcomote ITN will address them. The technical outcome will be a set of components of a single open platform named **Lowcomotive**. It will be a platform as a service, based on open standards at all levels (e.g., EMF, language-server-protocol, OpenStack). A frontend will provide LCE languages and smart interfaces that end-users will exploit to produce their artefacts. A backend will come in the form of a LCE repository providing server-side services for LCE developers. Industrial partners will provide case studies to ESRs develop and run their experiments.

Acknowlegments. This project has received funding from the EU Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 813884.

References

1. Clay Richardson and John R. Rymer. The forrester wave$^{\text{TM}}$: Low-code development platforms, q2 2016. Technical report, Forrester Research, April 2016.
2. Paul Vincent, Van Baker, Yefim Natis, Kimihiko Iijima, Mark Driver, Rob Dunie, Jason Wong, and Aashish Gupta. Magic quadrant for enterprise high-productivity application platform as a service. Technical report, Gartner, April 2018.
3. John Rymer R. and Clay Richardson. Low-code platforms deliver customer-facing apps fast, but will they scale up? Technical report, Forrester Research, August 2015.
4. Marco Brambilla, Jordi Cabot, and Manuel Wimmer. *Model-Driven Software Engineering in Practice, Second Edition*. Synthesis Lectures on Software Engineering. Morgan & Claypool Publishers, 2017.
5. Diego Vaquero-Melchor, Javier Palomares, Esther Guerra, and Juan de Lara. Active domain-specific languages: Making every mobile user a modeller. In *Proc. ACM/IEEE MoDELS*, pages 75–82. IEEE Computer Society, 2017.
6. Jon Whittle, John Hutchinson, and Mark Rouncefield. The state of practice in model-driven engineering. *IEEE Software*, 31(3):79–85, 2014.
7. Kerstin Altmanninger, Martina Seidl, and Manuel Wimmer. A survey on model versioning approaches. *IJWIS*, 5(3):271–304, 2009.
8. Petra Brosch, Gerti Kappel, Philip Langer, Martina Seidl, Konrad Wieland, and Manuel Wimmer. An introduction to model versioning. In *Proc. SFM*, pages 336–398, 2012.
9. Alexandra Mazak and Manuel Wimmer. Towards liquid models: An evolutionary modeling approach. In *Proc. 18th IEEE CBI, Volume 1 - Conference Papers*, pages 104–112, 2016.
10. Mark Buenen and Govind Muthukrishnan. World quality report 2017-18. Technical report, Capgemini, Sogeti, Micro Focus, 2017.
11. Michel Albonico, Stefano Di Alesio, Jean-Marie Mottu, Sagar Sen, and Gerson Sunyé. Generating Test Sequences to Assess the Performance of Elastic Cloud-based Systems. In *Proc. IEEE Cloud*, Honolulu, United States, 2017.
12. Dimitrios S. Kolovos, Richard F. Paige, and Fiona A.C. Polack. The grand challenge of scalability for model driven engineering. In *Models in Software Engineering*, pages 48–53. Springer, 2009.
13. Antonio Garmendia, Esther Guerra, Dimitrios S. Kolovos, and Juan De Lara. Emf splitter: A structured approach to emf modularity. In *Proc. eXtreme Modeling Workshop at MoDELS*, pages 22–31. 2014.
14. Dimitrios S. Kolovos, Louis M. Rose, Nicholas Matragkas, Richard F. Paige, Esther Guerra, Jesús Sánchez Cuadrado, Juan De Lara, István Ráth, Dániel Varró, Massimo Tisi, and Jordi Cabot. A research roadmap towards achieving scalability in model driven engineering. In *Proc. BigMDE*, pages 2:1–2:10. ACM, 2013.
15. István Ráth, Gábor Bergmann, András Ökrös, and Dániel Varró. Live model transformations driven by incremental pattern matching. In *Proc. of ICMT 2008*, pages 107–121, 2008.