Supporting Method Creation, Adaptation and Execution with a Low-code Approach

Raquel Araújo de Oliveira, Mario Cortes-Cornax, Agnès Front

Supporting Method Creation, Adaptation and Execution with a Low-code Approach

Raquel Araújo de Oliveira, Mario Cortes-Cornax, and Agnès Front

raquel.oliveira@univ-grenoble-alpes.fr,
mario.cortes-cornax@univ-grenoble-alpes.fr,
agnes.front@univ-grenoble-alpes.fr
Univ. Grenoble Alpes, CNRS, Grenoble INP*, LIG, 38000

Abstract. Method Engineering emerged in the 90s as a discipline to design, construct and adapt methods, techniques and tools for the development of information systems. By executing a method step by step, users follow a systematic and well-defined way to attain the results which the method was created for. To support the creation of methods in a more guided and systematic way, a method framework can be used as a template, allowing one to benefit from the expertise of method engineers who regrouped their good practices in such frameworks. However, the creation and adoption of a method may be difficult if there is no tool to support these activities. In addition, method engineers may not have the programming skills to implement such a tool. In this context, we propose an approach inspired by the *low-code* paradigm for Method Engineering. The approach helps method engineers in creating new methods or adapting an already existing framework that integrates some construction rules for guidance. Our approach automatically provides tool support so that method experts can actually execute the method. This paper presents the approach through a proof of concept implementation, and a first empirical evaluation through semi-structured interviews.

Keywords: Method Engineering · Low-code · Framework of Methods.

1 Introduction

In software engineering, development methods formalize the steps to follow in the software development life-cycle, looking for quality and client satisfaction. In [1] a method is defined as "*a process* for generating a set of models that describe various aspects of a software being built using some well-defined notation". Later on, this definition evolved to include not only the *process model* but also the *product meta-model*, to characterize the artifacts manipulated by the process.

Method Engineering is defined as the discipline to design, construct and adapt methods, techniques and tools for the development of information systems [2]. This engineering discipline promotes the adaptation of methods to particular

* Institute of Engineering Univ. Grenoble Alpes

contexts or projects. Particularly, our intent is to focus on one type of methods, namely *methods for the continual evolution or improvement* of different types of systems such as *ecosystems*, *business processes*, *production processes*, etc. Indeed, nowadays, businesses are living in an ever-changing environment, supported by technological, environmental and societal breakthroughs. This dynamism demands organizations to continually evolve to provide faster, better or more innovative products and services to stay competitive in the market.

To create a new method, one should be well-aware of the application domain as well as the Method Engineering field. To facilitate this task, frameworks have been proposed to serve as a template from which a new method can be created. Method engineers can be in charge of creating such frameworks. However, this task is not well guided and requires considerable experience [4]. In addition, although such frameworks provide a huge help in terms of creating a new method, little support is found afterward [4], on the usage of the method.

Once a method is created for a specific need, an efficient execution of a method requires some tool support. Indeed, each step of the method's process can be handled using a variety of different techniques, and using different tools. For instance, in the Agile Methodology, in particular the Scrum Process, we can mention two steps needing different techniques: the definition of the product vision could be done using *brainstorming*, while following up the work could be done by *daily meetings*, and specific tools to support these techniques exist. While it is possible to execute the method step by step without a centralized control, and to somehow manually keep track of where we are and what was already done, it is more convenient to centralize this information. A tool to support method creation, adaptation and execution would therefore facilitate a method application. However, method engineers do not always have software development skills to create such a tool. A *low-code* approach (LCA) can help in achieving this goal, by automating the creation of a tool to support the execution of the created method. Indeed, the low-code paradigm promotes the development of applications with little coding, to rapidly deliver applications [20]. Our research question is therefore : *How can Method Engineering benefit from a low-code approach in order to create, adapt and execute methods ?*

In [17] we presented a preliminary work where low-code was applied to support method creation and execution. In this paper, we extend the previous one by integrating the possibility to create frameworks of methods (from which methods can be created by adaptation), thereby including a third profile of actors who can benefit from the approach: framework engineers (cf. Fig 1). Hereafter, we present the approach and the implemented prototype, specially the method executor module, generated automatically through the *low-code* engine. We also report on preliminary user experiments performed on the prototype.

The outline of the paper is the following. First, Section 2 gives background and the state of the art relying on Method Engineering and low-code approaches. Section 3 overviews our low-code approach to support Method Engineering, followed by Section 4 which relies on an example to illustrate the implemented prototype. Section 5 presents the implementation as a proof of concept of the

approach. Section 6 presents the first steps of a user-centred validation. Finally, Section 7 concludes and highlights some future work.

2 Background and State of the Art

Approaches proposed in **Method Engineering** (ME) for the construction of a method are generally classified in [14, 18]:

- *ad-hoc* approaches concerning the construction of a method "from scratch";
- *assembly-based* approaches proposing to reuse and compose method components (fragments, "method chunks", services, ...) to construct a specific method or a method family;
- *extension-based* approaches consisting of extending a method to produce a new one;
- *model-based* or *paradigm-based* approaches, where the construction of a new method is realized by instantiation or model adaptation [9].

We focus on a model-based approach by adaptation, where the construction of methods is based on the adaptation of methods of the same abstraction level. In this kind of approaches, methods are defined by both a *process model* and a *product meta-model*, relying on the OMG meta-layers of models [12]. The process model details the steps to follow to accomplish the goal of the method, while the product meta-model describes the artifacts manipulated by the process. Both the process model and the product meta-model are positioned in different meta-layers. The process model conforms to an existing language (i.e. *Maps*, presented in Section 4), while the product is positioned at the meta level.

Nowadays, method engineers must deal with vast and variable contexts when they are in charge of putting in place a method guiding the continual evolution of a system. Different examples of continual evolution methods can be found in the literature, each one proposing its own cycle of steps and ways to characterize the artifacts guiding the evolution, for example, PDCA (Plan, Do Check, Act) [8], DMAIC (Define, Measure, Analyze, Improve, Control) [7], and A3 [22]. Except for DMAIC [7], the aforementioned methods have not been formalized, which makes it challenging to integrate them in a framework of methods.

Different approaches [4, 2, 9, 18, 14] have been proposed in ME to deal with the construction of new methods by adapting existing ones. However, for the best of our knowledge there are no tools to support the creation of methods based on these approaches, and such adaptation is mainly paper-based. We formalized in previous works the As-Is/As-If framework [4] as a conceptual framework dedicated to the family [9] of continual evolution methods. This framework is composed of a process model and a product meta-model that method engineers can adapt when constructing a target method. The last one results in a method guiding the continual evolution of a system in a given context.

In our previous works, the use of this framework was simply guided by heuristics that method engineers could "manually" apply to adapt the process model and the product meta-model to their own domain. This paper presents a novel

approach based on the low-code paradigm, to help method engineers to construct a new method, by adaptation of a framework of methods (such as As-Is/As-If), and to follow its execution.

Low-code development platforms (LCDP) enable rapid application delivery with a minimum of hand-coding, and quick setup and deployment [20]. The goal of such platforms is to ease the development of applications, by reducing drastically the coding efforts, or by eliminating coding all together. Although the term *low-code* first appeared in the literature in 2014 [20], some principles behind it are not completely new. There have been efforts to easy applications' development since a long time ago. Very popular in the 90s, Microsoft Office Access provides an easy way to create simple desktop applications with input forms to manage data in information systems. WordPress[1] has also been in the market since the 2000s, providing a user-friendly way to create websites. The main difference with LCDP is that a bigger focus has been made on fast application delivery, beside the rapid application development (usually based on a visual and drag'n'drop way of development).

However, for the best of our knowledge, no LCDP had yet focused on ME. Most LCDP can be classified as *general purpose* platforms, while some are specific to *process modeling*. According to [19], general-purpose LCDPs target a wide range of applications with tools that address application creation, integration, deployment, life-cycle management, and distribution. Some general-purpose LCDP propose workflow automation, which to some extent relates with the method execution we propose to support in this paper. Examples of such LCDP are Mendix[2], OutSystems[3], and Simplicité[4]. However, the workflow automation in these LCDP helps in defining the business logic of applications behavior, which may be adapted to methods processes, but it is not their main focus.

On the other hand, LCDPs specific to process modeling focus on creating and configuring process flow logic and workflows, specially business processes. Some of those LCDPs are AgilePoint[5], Appian[6], Bonitasoft[7], Activiti[8], JBoss jBPM[9] or Intalio[10]. Most of these LCDPs support the Business Process Model and Notation (BPMN 2.0) language [11] to represent processes and also provide constrained ways to define a data model of the process (e.g. relational databases or java classes). The main difference with our approach is that ours allows method engineers to define data at the meta level (i.e. the product meta-model). This provides the possibility to benefit from a *Domain Specific Language*, to guide the creation of the product model, giving more flexibility when defining the artifacts

[1] https://wordpress.com/
[2] https://www.mendix.com/
[3] https://www.outsystems.com/
[4] https://simplicite.fr
[5] https://www.agilepoint.com
[6] https://appian.com/
[7] https://www.bonitasoft.com/
[8] https://www.activiti.org/
[9] https://www.jbpm.org/
[10] https://www.intalio.com/

that may be used in the method. Besides, our approach allows a larger range of profiles (namely framework engineers, method engineers and method experts, cf. Fig 1) to benefit from the low-code paradigm. The next section introduces the approach and briefly presents the research methodology we followed.

3 Overview of a LCA to Support Method Engineering

This section provides an overview of a low code approach (LCA) focusing on the different users that can work together in the ME field. Based on our previous experience [4], we identified people with different profiles working together and playing different roles in the field of Method Engineering: (i) *framework engineers* know and can design different method frameworks from which methods can be created. These frameworks can be considered as method templates that may capitalize knowledge; (ii) *method engineers* are well aware of the Method Engineering discipline and can create a new method or adapt an existing framework or method for a specific purpose; finally (iii) *method experts* have deep knowledge in one (or more) specific method(s) and are able to lead its (or their) execution. The approach presented in this paper aims at supporting these three roles in achieving their goals.

Firstly, a framework engineer creates a framework of methods (Fig. 1), which facilitates further creation of a method. Such a framework would define a generic process model and product meta-model, adaptable to many application domains. It could include a set of heuristics, in order to restrict and guide the method con-

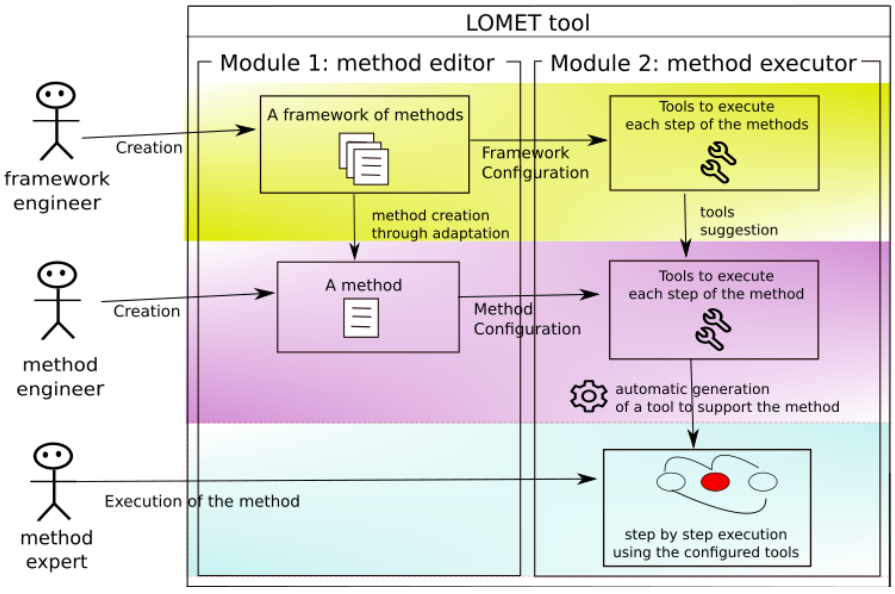

Fig. 1. A low-code approach to support Method Engineering.

struction rules. In the next step, framework engineers can configure the framework, by setting the most suitable tools to execute each step of the methods based on their framework.

Once a framework is created and configured, method engineers can create a method by adapting the latter, meaning that the process model and product meta-model of the framework can be extended to create a new method for a specific purpose, following the pre-defined framework heuristics. Alternatively, the method engineer can also create a method from scratch. The list of configured tools in the framework are then suggested to configure the new-created method (in case the latter was created based on a framework). In this step, a method engineer could also propose new tools.

Once a method is created and configured, the method expert can lead its execution step by step. Following the low-code paradigm, a supporting tool to execute the method is automatically generated and can be used by method experts in order to track the progression and the different artifacts resulting from each step.

In this work, we follow the THEDRE research methodology [16], a framework of scientific research in computer science. This method proposes guides to manage research, to correctly initiate the work, to experiment, to analyze data, and to write an article. This method is user-centred, therefore users are quickly put in the research cycle. In this research, the prototype we have developed to illustrate the approach was quickly evaluated by users, and the preliminary results are discussed in Section 6.

4 Running Example

This section presents the CEFOP [5] method as a running example which we will use to illustrate the approach presented in this paper, in Section 5. Fig. 2 depicts the usage of the As-Is/As-If framework for the creation of the CEFOP method, a method dedicated to continual evolution of business processes within small and medium companies. The goal of the CEFOP method is to incite process participants empowerment and collective decision-making on the possible evolutions, by making participants' intervention more objective in particular through traces analysis and process mining techniques.

On the As-Is/As-If framework, the product meta-model is represented using UML class diagrams (depicted on the left part of Fig. 2), a well-known standard notation for representing conceptual data models, while the process model is represented using the Map [21] language (depicted on the right part of Fig. 2), a well-known notation to represent goal-oriented processes. Indeed, the As-Is/As-If framework adopts an *intentional-based* approach, in which the process model promotes a clear difference between *what* to achieve (the intentions, represented by ellipses) and the *ways* to achieve it (the strategies, arcs linking the ellipses) [21]. On the other hand, the product meta-model defines all the concepts (data artifacts) manipulated by the process model, as well as their relationships. For instance, in the CEFOP method, continual evolution of

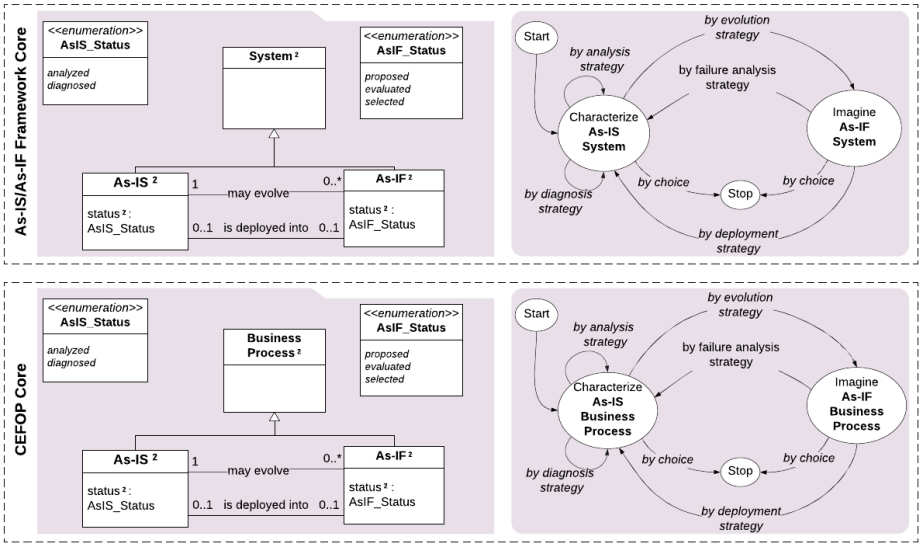

Fig. 2. As-Is/As-Is Core Package and its adaptation in CEFOP (on the left the product meta-models and on the right the process models).

business processes is the main focus, so in the product meta-model, a class to characterize business processes can be found (cf. the bottom-left part of Fig. 2).

Fig. 2 shows the adaptation of a fragment of the As-Is/As-If framework (the Core Package) to the CEFOP method. The top of Fig. 2 shows the template proposed by the As-Is/As-If framework for generic continual evolution methods, where we start from a current version of a system (As-Is) towards possible evolutions of it (As-If). The bottom of the image shows the CEFOP method adapted from the As-Is/As-IF template.

In this paper, we propose an approach to create frameworks of methods, allowing methods to be created or adapted in a more guided way. Furthermore, methods can be executed step by step. We use the As-Is/As-If framework to illustrate the creation of frameworks in our approach, nonetheless the approach is generic and can be used to create other frameworks of methods. The next section presents `LOMET` , a prototype that implements the approach.

5 Approach Implementation: the `LOMET` Tool

`LOMET` (for LOw-code Method Engineering Tool) is a web-based platform composed of two main modules: a *method editor* and a *method executor*. The method editor is a classic web diagram editor (dedicated to framework engineers and method engineers) allowing frameworks and methods to be created, while the method executor (dedicated to method experts) provides a supporting tool to execute a method, by applying the low-code paradigm (with user-friendly config-

urations and easy tool creation). Concerning the technologies, JointJS[11], HTML and CSS are used in the front-end, and Express[12] and MySQL in the back-end. Following, the aforementioned modules are described in more detail.

5.1 Creating a Framework of Methods (or a Method)

Fig. 3 illustrates the framework of methods (and method) editor. At the top of the web editor, the method engineer or the framework engineer can choose to create a method or a framework. Several functionalities are available in both cases, accessible through a toolbar: *Save as* (both in a JSON file and in a database) and *Open* (from a JSON file), *Associate* chunks of the product meta-model with a part of the process model, and *Save* the method *as an image*. In case we want to create a method, through the button *From framework* we can select which framework our method is based on. There is also the possibility to skip this and create a method from scratch. Using the toolbar, we can also indicate that we want to apply the framework heuristics when creating our own method, or on the contrary if it is preferable to freely create our method using the framework as a template. Note that definition and use of the heuristics are out of the scope of this paper. The reader can refer to [4] for a deepen explanation. Other features available on the left and right sides of the user interface are the *Visual Zoom* and *Open a framework/method from the database* (a blue engine button). The Visual Zoom selector allows one to alternate the mouse scroll button function between visually zoom in/zoom out the diagrams and simply scroll the webpage.

The editor main zone contains two pages for creating the process model and the product meta-model (using Maps and UML class diagrams) of a method or a framework. A sticky toolbar available on each side of the pages allows one to drag'n'drop visual elements inside the models. We can connect elements by creating a link from a source to a target element. For instance, the "by evolution strategy" on the process model which connects the "Characterize As-Is Business Process" and "Imagine As-If B. Process" intentions (Fig. 3), or the associations between the "AS-IS" and "AS-IF" meta-classes on the product meta-model.

The process model includes an interesting feature called *Semantic Zoom* [10], a graphical technique to balance detail and context. While a physical zoom changes the size and visible details of objects, a semantic zoom changes the type and meaning of information displayed by the object, by using a semantic transition between detailed and general views of information. In `LOMET`, by clicking in a strategy, one can detail how it can be subdivided, in another sub-map below the previous one. For instance, in Fig. 3, the "by analysis" strategy (in dashed line) used to iteratively perform the intention "Characterize As-Is Business Process" can be further detailed in a sub-map, with extra intentions and strategies. Furthermore, the "by specification" strategy to go from the "start" to the "Identify Process Components" intentions is also detailed in a second sub-map below the previous one. Two levels of semantic zooms are available.

[11] https://www.jointjs.com/
[12] https://expressjs.com/

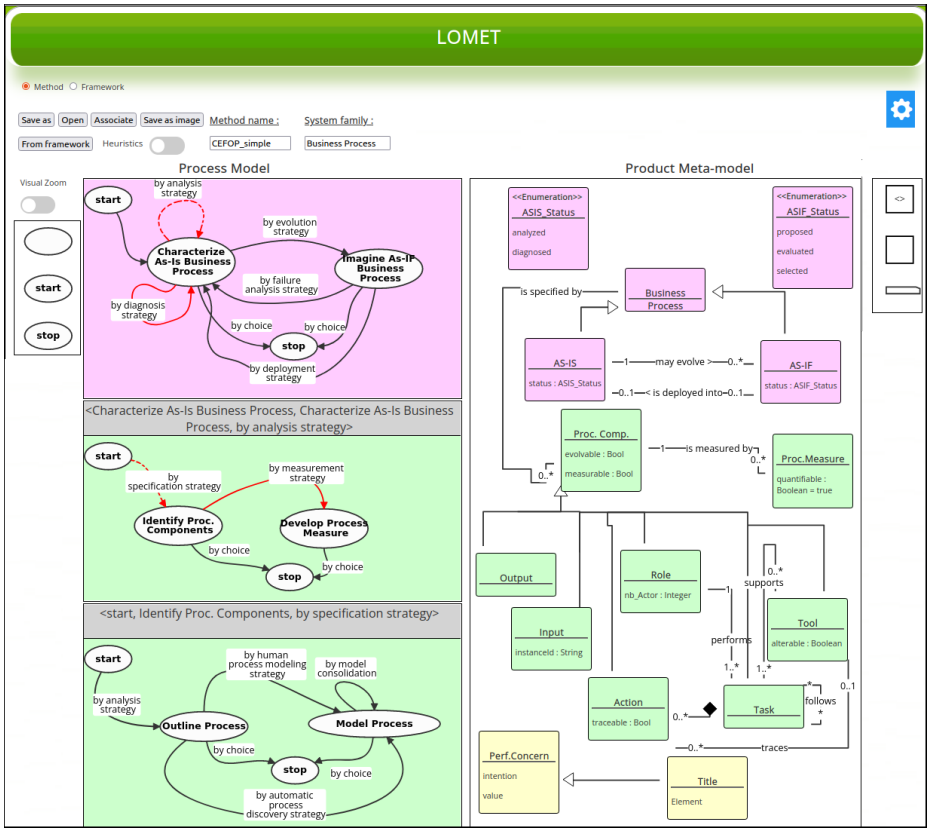

Fig. 3. User interface to create a framework of methods and a method in LOMET.

While the process model of a method is progressively created from a generic view (the main Map) to more detailed ones (the two sub-maps), the product meta-model is a single model containing all the elements manipulated by the process model in the context of the method. To identify more precisely which meta-classes are manipulated by a process model's fragment, we can associate them in LOMET using the *Associate* button in the toolbar. By linking a strategy to a set of meta-classes, LOMET colorizes them equally, resulting in the visual render illustrated in Fig. 3. Here, the "Process Component" (and its subtypes) and the "Process Measure" meta-classes are manipulated by the "by analysis" strategy in the process model (detailed in the two lower sub-maps).

5.2 Configuring a Framework of Methods (or a Method)

Once the framework of methods (or the method) is created, the framework engineer (or the method engineer) can configure it by indicating the most suitable

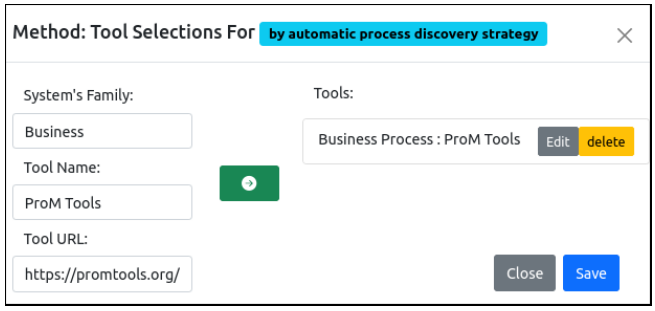

Fig. 4. User interface to configure a method in LOMET.

tools to execute each step of the method (cf. Fig. 1). This is done at the process model level, and to do so, the method engineer selects each strategy of the process model, and sets the URLs to the most suitable external tools to execute this part of the method. For instance, in Fig. 3, in order to "Identify the Process Components" at the CEFOP method (an *intention* in the second map), the first thing to do is to "Outline the Process", followed by "Model the Process" (in the third map), and one possible strategy to outline the process is "by automatic process discovery" (in the third map). This can be done using process mining tools such as ProM [13]. Therefore, the framework engineer (or the method engineer) can suggest this tool in LOMET (cf. Fig. 4), so that the method expert uses it when executing this step of the method.

In its current state, the method execution only takes into account the steps defined in the process model. The product meta-model is not yet exploited, so it is only informative. We already included a way to create product meta-models in the method editor in order to exploit them when the method is executed, as a future work. For instance, a product model could define the elements (process components) in which the business process improvement will focus on, such as inputs, outputs or tasks (cf. the product meta-model in Fig. 3), ignoring for instance activity execution flows. Therefore, when outlining a business process, only the elements identified in the product model will be considered in its continual evolution.

5.3 Executing a Method

Following the low-code paradigm, LOMET automatically generates a supporting tool for method experts to execute the method created and configured beforehand. The advantages of the supporting tool are threefold: firstly, method experts are provided with a visual indication of which steps of the method were already executed (in green), which one they are currently on (in yellow), and which ones are pending (in red), cf. Fig. 5. Currently, method experts indicate manually in the tool the status of each step, by clicking on a strategy and changing its

[13] https://promtools.org/

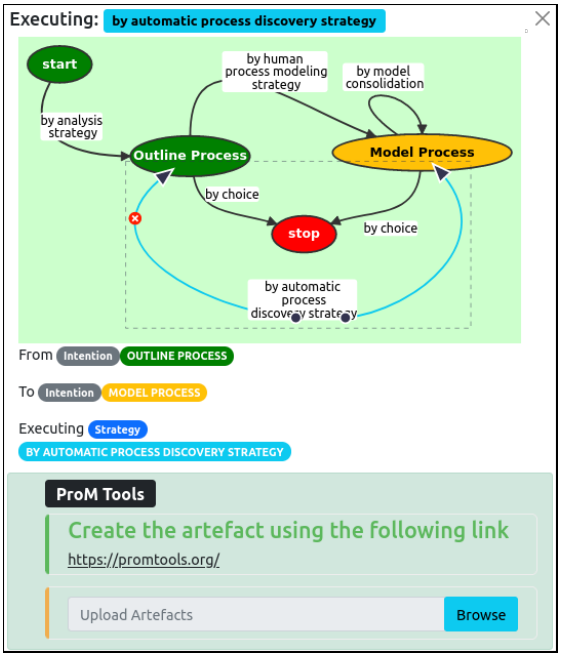

Fig. 5. User interface to execute a method in LOMET.

status. Secondly, method experts can execute the method step by step using the external tools configured by the method engineer. And lastly, they can keep in LOMET a repository of artifacts produced during the method execution.

To execute the method in the generated supporting tool, method experts: (i) click on each strategy of the process model (the top part of Fig. 5); (ii) click in the URL set by the method engineer (the bottom part of Fig. 5); (iii) are redirected to the external tool; (iv) where they can create an artifact in this external tool (for instance, a PDF containing the processes automatically discovered by process mining using the ProM tools); and (v) import this artifact back into LOMET (button *Browse* in Fig. 5). They can then follow these steps for each strategy of the process model in order to execute the method. If the method expert has already imported an artifact for a strategy before, she/he can visualize it in the same user interface (just below the *Browse* button in Fig. 5), allowing her/him to keep in LOMET a repository of artifacts produced by the method execution, for further access.

Our low-code prototype proposes a unique tool to execute the method step by step, helping method experts to handle the heterogeneity of existing tools suitable for each strategy of a method process model. Method experts may not be aware of these tools, and the method engineer expertise is necessary to configure the method with the most suitable tools. The low-code paradigm finds here an interesting application in the Method Engineering domain, allowing method

engineers to provide a supporting tool for method experts to adopt the method without having to actually implement the tool.

6 Validation of the Functional Requirements of LOMET

This section presents an evaluation in progress of LOMET . The aim is to verify if the tool provides the adequate functional requirements for method engineers to build and provide support to their own methods. We also evaluate the tool usability. Non-functional requirements of LOMET are out of the scope of this paper. A user-centred experiment has been put in place within researchers that participated in the creation of their continual evolution methods without the help of any tool.

To carry out the evaluation, we choose a method recommended by sociologists and also by computer designers: semi-structured interviews [13]. The interviews are completed by observing the participants. We choose this method as it helps to identify ideas, opinions, or habits. These interviews are organized face to face with an interview schedule grid. For the moment, only three interviews have been realized, with research colleagues that lead projects where continuous evolution methods were produced (ADInnov [6], CEFOP [5] and CircusChain[15]).

We rely on our running example using the As-Is-As-If framework to provide to the subjects an incomplete adaptation of their own methods. After a video demonstration, the subjects had to complete a part of their methods through a protocol with different steps : 1) import the template; 2) manipulate the process model in order to complete a strategy using a semantic zoom; 2) manipulate the product meta-model in order to define the data model (product meta-model) manipulated by the process model; 3) configure the method to define the appropriate tools to execute the strategy; and finally 4) execute the defined strategy including the upload of a document representing the artifact of the aforementioned strategy execution using the pre-defined tool.

The validation had to produce some evidence regarding the following 3 usefulness/usability statements : 1) *"The LOMET tool is an appropriate way to monitor the execution of a method"*; 2) *"The LOMET tool is helpful to create a method compared to the creation without a tool"*; and 3) *"The LOMET tool is easy to use"*. The available material was: 1) the case study containing the exercise description; 2) two questionnaires explained below; 3) a 10-min video demo; and 4) the LOMET tool, accessible via a (for the moment) local URL.

The **first questionnaire** targeted the profile description of the subject. We question the subjects about what they consider to be their knowledge in the Method Engineering topic, what are their potential difficulties that they faced when building their methods without tool support, the functionalities that they may want to have in order to support their methods and how they currently follow up the method execution. Some interesting points follow. Firstly, none of the interviewed experts had a way to configure nor execute their methods, what they consider as drawbacks for the method adoption. Secondly, some desired functionalities were indicated: 1) the versioning possibility; 2) the possibility

to use method fragments to build a new method; 3) a way to trace the method execution with artifacts on each step; 4) the possibility to highlight the coherence between the process model and the manipulated artifacts; 5) the possibility to highlight the impacts of a change in the method; and 6) the possibility to instantiate the method. As shown in the previous sections, all the mentioned desired functionalities are already considered in `LOMET` , apart from the impact analysis. The fragment composition is not yet well-developed but the possibility to define some fragments in the form of a framework, which can be then imported to create a new method, partly supports this feature.

After the video visualization and the exercises, the **second questionnaire** was used in order to ask the subjects about the advantages and disadvantages of the tool, as well as the difficulties that `LOMET` could have minimized when building their own methods. We also asked about the appropriateness of the proposed functionalities, and we rely on Brooke's System Usability Scale (SUS) [3] to evaluate the tool's usability.

The following **advantages** were put forward: 1) the tool was considered an appropriate way to enable knowledge capitalization; 2) the possibility to import a framework (template) accelerate the method construction; 3) the fact of using a specific and constrained language for modeling the method was also appreciated; 4) the process model was considered easy to manipulate. The semantic zooms helped to master the potential complexity; 5) the possibility to assign dedicated tools to the process strategies at the framework and method creation level was considered useful; 6) the use of colors and the possibility to associate process model elements with the product meta-model elements was also appreciated.

The following **disadvantages** were highlighted: 1) the method execution was considered limited (mainly status update and artifact uploads). Also, no automatic status update of the intention was proposed; 2) several usability problems were detected such as the need for clarification of the framework import and opening, the difficult way to add associations in the product meta-model, the non-intuitive image export, the lack of standardized way to interact with graphical elements, or the lack of a Ctrl+Z option. Nevertheless, considering the SUS questionnaire, the answers were mostly positive; 3) the product meta-model manipulation was sometimes hard, especially if there is a need to manipulate large models. There is no way to manage the complexity using semantic zooms as in the process model part; 4) there is no easy way to re-use method fragments. Indeed, even if the tool provides the possibility to reuse a framework as a template, there is no dedicated functionality to compose method fragments.

Considering the aforementioned problems, we plan to integrate the corresponding **improvements**. In a short term period, we aim at treating the execution limitations and enrich it with a specific forum so users can comment each uploaded artifact. In addition, the fact of adding a protocol that may be attached to the tool in order to guide its usage is also a priority. The prototype nature of the tool indeed induces some usability limitations that we also aim to treat rapidly. In a mid-term period, we want to explore the possible integration with a dedicated UML editor in the tool in order to facilitate the class diagram edition,

for the product meta-model. Finally, the introduction of method fragments is considered to be a long-term objective that still has to be studied.

To conclude, it is still soon to consider that our three usefulness/usability statements have been validated, as we have only performed three interviews. Still, as we show here, we have already received feedback that is useful to guide future improvements of the tool and the general approach. Indeed, the user-centred interviews are a valuable way to evaluate and improve the approach, with promising results.

7 Conclusion

We have presented in this paper an approach to support the creation of frameworks of methods, from which one can derive methods adapted to her/his domain. The approach benefits from the advantages of the low-code paradigm, allowing method engineers, who may not have programming skills, to produce and configure a tool to support method experts on the execution of methods. Therefore, we have tackled our research question presented in Section 1.

In our approach, a method is formalized by means of a process model (represented by Maps) and a product meta-model (represented by a UML class diagram). Any framework of methods using these formalisms can benefit from our approach. Once a method is created, the method engineer configures the most suitable tools to execute each step of the method, from which method experts can execute the method using a tool derived from this configuration, without having to write one single line of code.

As mentioned in Section 6, we plan at extending the tool by adding the possibility to include a set of guidelines (i.e. protocols) in the method. These protocols could be defined by the method engineer, in order to better guide the method expert when executing the method. Besides, we plan to extend the configurations' possibilities that method engineers do in the method, since for the moment they can set a list of tools allowing the method expert to execute each step of the method.

Acknowledgements This work is funded by the Emergence Projects 2022 of the *Laboratoire d'Informatique de Grenoble* (LIG). We thank the work of Noe Choc, Junhao LI and Aymeric Surre in helping with the development of `LOMET`.

References

1. Booch, G.: Object oriented design with applications. Benjamin-Cummings Publishing Co., Inc. (1990)
2. Brinkkemper, S.: Method engineering: engineering of information systems development methods and tools. Information and software technology **38**(4), 275–280 (1996)
3. Brooke, J., et al.: SUS-a quick and dirty usability scale. Usability evaluation in industry **189**(194), 4–7 (1996)

4. Çela, O., Cortes-Cornax, M., Front, A., Rieu, D.: Methodological framework to guide the development of continual evolution methods. In: Advanced Information Systems Engineering: 31st International Conference, CAiSE 2019, Rome, Italy, June 3–7, 2019, Proceedings 31. pp. 48–63. Springer (2019)
5. Çela, O., Front, A., Rieu, D.: Cefop: A method for the continual evolution of organisational processes. In: 2017 11th International Conference on Research Challenges in Information Science (RCIS). pp. 33–43. IEEE (2017)
6. Cortes-Cornax, M., Front, A., Rieu, D., Verdier, C., Forest, F.: Adinnov: An intentional method to instil innovation in socio-technical ecosystems. In: Advanced Information Systems Engineering: 28th International Conference, CAiSE 2016, Ljubljana, Slovenia, June 13-17, 2016. Proceedings 28. pp. 133–148. Springer (2016)
7. Deeb, S., Bril-El Haouzi, H., Aubry, A., Dassisti, M.: A generic framework to support the implementation of six sigma approach in smes. IFAC-PapersOnLine **51**(11), 921–926 (2018)
8. Deming, W.E.: Out of the Crisis, reissue. MIT press (2018)
9. Deneckere, R., Kornyshova, E., Rolland, C.: Method family description and configuration. In: ICEIS. pp. 384–387 (2011)
10. Garcia, J., Theron, R., Garcia, F.: Semantic zoom: A details on demand visualisation technique for modelling owl ontologies. In: Highlights in Practical Applications of Agents and Multiagent Systems: 9th International Conference on Practical Applications of Agents and Multiagent Systems. pp. 85–92. Springer (2011)
11. Group, O.M.: Business process model and notation - (bpmn 2.0) (2011), http://www.omg.org/spec/BPMN/2.0
12. Group, O.M.: Meta object facility (2016), https://www.omg.org/spec/MOF/About-MOF/
13. Hindus, D., Mainwaring, S.D., Leduc, N., Hagström, A.E., Bayley, O.: Casablanca: designing social communication devices for the home. In: Proceedings of the SIGCHI conference on Human factors in computing systems. pp. 325–332 (2001)
14. Hug, C., Front, A., Rieu, D., Henderson-Sellers, B.: A method to build information systems engineering process metamodels. Journal of Systems and Software **82**(10), 1730–1742 (2009)
15. Kurt, A.: Models and tools for the design, assessment, and evolution of Circular Supply Chains. Master's thesis, University of Grenoble (December 2021)
16. Mandran, N.: THEDRE: langage et méthode de conduite de la recherche: Traceable Human Experiment Design Research. Ph.D. thesis, Université Grenoble Alpes (ComUE) (2017)
17. Oliveira, R., Cortes-Cornax, M., Front, A., Demeure, A.: A low-code approach to support method engineering. In: Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings. pp. 793–797 (2022)
18. Ralyté, J., Rolland, C., Ayed, M.B.: An approach for evolution-driven method engineering. In: Information Modeling Methods and Methodologies: Advanced Topics in Database Research, pp. 80–101. IGI Global (2005)
19. Richardson, C., Rymer, J.R.: Vendor landscape: The fractured, fertile terrain of low-code application platforms. FORRESTER, Janeiro (2016)
20. Richardson, C., Rymer, J.R., Mines, C., Cullen, A., Whittaker, D.: New development platforms emerge for customer-facing applications. Forrester: Cambridge, MA, USA **15** (2014)
21. Rolland, C.: Capturing system intentionality with maps. In: Conceptual modelling in Information Systems engineering, pp. 141–158. Springer (2007)
22. Shook, J.: Managing to learn: using the A3 management process to solve problems, gain agreement, mentor and lead. Lean Enterprise Institute (2008)