How to cite: Mosquera, D., Martakos, A., Ruiz, M. (2022). Experiences from Developing a Web Crawler Using a Model-Driven Development Tool: Emerging Opportunities. In: Augusto, A., Gill, A., Bork, D., Nurcan, S., Reinhartz-Berger, I., Schmidt, R. (eds) Enterprise, Business-Process and Information Systems Modeling. BPMDS EMMSAD 2022 2022. Lecture Notes in Business Information Processing, vol 450. Springer, Cham. https://doi.org/10.1007/978-3-031-07475-2_23

Experiences from Developing a Web Crawler using a Model-driven Development Tool: Emerging Opportunities

David Mosquera[0000-0002-0552-7878] ✉, Anastassios Martakos, and Marcela Ruiz[0000-0002-0592-1779]

[1] Zürich University of Applied Sciences, Gertrudstrasse 15, Winterthur 8400, Switzerland
{mosq, ruiz}@zhaw.ch, martaana@students.zhaw.ch

Abstract. Model-driven development (MDD) tools aim to increase software development speed and decrease software time-to-market. Available MDD tools in the market state that software development teams can fast and easily develop "any" software by using them. So, the following research question arises: what is the perception of a software developer in using an MDD tool to create software he/she is used to develop without models? We selected Mendix, a user-friendly and easy configurable MDD tool, to address such a question and develop a domain-specific software artifact. We propose a use case collaborating with a Swiss company that allows users to compare insurances based on web crawling. Therefore, we ask a software developer at the Swiss company to develop a simplified version of a web crawler using the selected MDD tool. The software developer has extensive experience with developing web crawlers. However, for the software developer using MDD tools was a new paradigm of software development. We observe that the software developer successfully developed the web crawler using the MDD tool. However, he/she perceived some difficulties during the development, arising opportunities such as decreasing modeling complexity, increasing the MDD tool integrability, and improving modeling assistance. Finally, we conclude the experience report by drawing next research endeavors to generalize the results and discover new opportunities for improving MDD tools.

Keywords: Model-driven development, Web crawling, Experience report.

1 Introduction

Model-driven development (MDD) tools promise to increase the productivity of software development teams and decrease software time-to-market [1]. Thus, software development teams invest their effort in creating conceptual models that describe the under-development software application rather than coding. Then, an MDD tool allows them to automatically transform such conceptual models into code using automatic model-to-model and model-to-text transformations.

Several MDD tools are available in the market [2–6] and literature [7–10] aiming to achieve the MDD "promise." These tools state that software development teams can

fast and easily develop software by using them. Based on such a statement, a software developer should be able to develop most of the functionalities as he/she uses to develop using other approaches. So, the following research question arises: what is the perception of a software developer in using an MDD tool to develop software he/she is used to developing without models?

We design a use case collaborating with a Swiss company to address our research question. This company allows users to compare insurance premiums from several insurance providers using web crawling. Web crawling allows for automatically extracting data from websites using software [11]. As a result, web crawlers gather details from web pages in near real-time to present them to the user in a single source summarized way. Developing web crawlers require domain-specific knowledge, making it a relevant domain for addressing our research question. Moreover, we select Mendix—one of the MDD tool market leaders—to develop the use case [6] and test our research question.

We ask an experienced software developer at the Swiss company to develop the designed use case using Mendix. The software developer has extensive experience with developing web crawlers. However, for the software developer using MDD tools was a new paradigm of software development. We document all the use case development, observing that the software developer successfully developed a simplified web crawler using Mendix. However, he/she perceived some difficulties using Mendix. Based on such remarks, we identified three main opportunities for improvement: decreasing modeling complexity, increasing MDD tool integrability, and improving modeling assistance. These identified opportunities for improvement are helpful and relevant to model-driven engineers—i.e., who develop MDD tools—although more extensive data gathering is required to validate and generalize the results. In future work, we expect to replicate this experience with several web crawling and other domain software developers to validate and improve the identified opportunities.

This experience report is structured as follows: in Section 2, we review the available MDD tools in the market and we motivate our research question; in Section 3, we introduce the use case designed in collaboration with the Swiss company; in Section 4, we report the results and the identified opportunities for improvement based on the software development perception using Mendix; and, finally, in Section 5 we discuss conclusions and future work.

2 MDD tools overview and motivation

Software specifications are created by conducting analysis, requirement specification, and design. Conceptual models are used throughout this process to represent the software under-development. In code-centric approaches, software developers manually turn these conceptual models into code. MDD tools propose to go one step further by using the models as blueprints to automatically generate the code based on such conceptual models [1, 12, 13]. As a result, software development teams improve their productivity and decrease the software time-to-market.

To support such a transformation process, several MDD tools have been developed and are now available in the market to develop software based on conceptual models. For instance, OutSystems [2] offers a domain-specific language (DSL) as a fourth-generation language that provides a graceful fullback to third-generation programming languages such as C#. Microsoft PowerApps [3] allows companies to create software using a drag-and-drop editor, integrating such software into the Microsoft ecosystem. Appian [4] enables businesses to automate their processes, producing mobile-ready applications integrated into cloud systems without programming. WebRatio [5, 14] offers an Eclipse-based [15] developing environment for creating web and mobile software applications by using IFML (Interaction Flow Modeling Language) models. Finally, Mendix [6] uses a visual editor with its own modeling language to represent business logic workflows, generating web-based applications.

We observe these MDD tools state that "any" software can be developed by using them, making statements such as: "anyone with an idea can make powerful apps [6]," "we accelerate customers' business by discovering, designing, and automating their most important processes [4]," "automate your business processes and bring them online [5]," among others. We observe that the MDD tools delimit the "any" software idea to specific domains, such as business processes. However, the MDD tools promise that software developers can develop the software they are used to developing using models in such specific domains. Therefore, the following research question arises:

(RQ) What is the perception of a software developer in using an MDD tool to develop software he/she is used to develop without models?

3 The use case: Web crawling and MDD tools

We plan to ask a software developer to use one of the MDD tools reviewed in Section 2 and develop a domain-specific use case to address our RQ. We had the opportunity to collaborate with a Swiss company that offers online services for comparing insurances based on web crawling. Therefore, we select web crawling as our domain-specific use case. We introduce such a use case in the following paragraph.

The Swiss company uses web crawling to automatically retrieve data about insurances, allowing users to compare them in a single source summarized web page. The web pages where the Swiss company extracts data are named *targets* [11]. Sometimes, the web crawler needs to be rewritten when a new *target* appears. Therefore, the Swiss company proposes crawling new *targets* as soon as is required as the use case. As a result of developing this use case, the web crawler must collect the data from such a new *target* and store it in the database for further processing, as shown in Fig. 1.

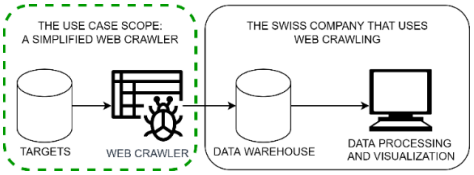

Fig. 1. Use case overview.

Having proposed the use case, the next step is selecting an MDD tool to develop it. As a proof of concept, we selected Mendix from the reviewed MDD tools since it is a user-friendly tool that allows software developers to kick-start modeling quickly. We considered other MDD tools; however, they were not that easy to configure and run the software as Mendix. For instance, Mendix has a web environment that allows software developers to use the tool as soon as they login into the Mendix website, without any additional configuration. Other tools, such as WebRatio, require the software developers to download an IDE (Integrated Development Environment) software and configure external elements such as databases and web servers, hindering their configuration. Moreover, Mendix is one of the MDD tool leaders in the market based on the Gartner Magic Quadrant (see Fig. 2).

Fig. 2. Gartner Magic Quadrant on enterprise low-code application platforms, taken from [16].

4 Results on developing the use case

We ask a software developer from the Swiss company to use Mendix to develop the use case introduced in Section 3. The software developer has more than four years of industrial experience, working two years at the Swiss company. The software developer has enough expertise to develop a web crawler, making him/her a feasible subject to create a web crawler using Mendix. The software developer invested approximately 40 working hours in developing the use case using Mendix, including learning how to use the tool itself since it was her/his first time using an MDD tool. As a result, the software developer created: a domain model containing the information of the web crawler targets (see Fig. 3); a set of microflow models comprising the business logic (see Fig. 4); and a graphic user interface for managing the web crawler targets (see Fig. 5).

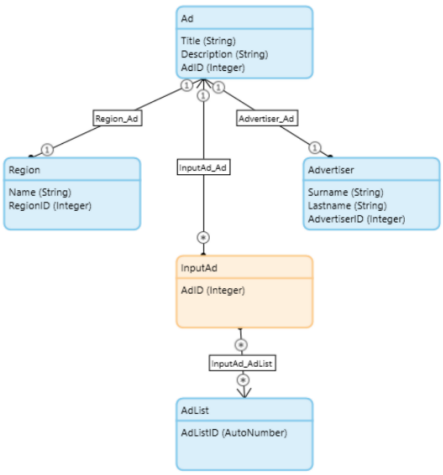

Fig. 3. Web crawler Mendix's domain model.

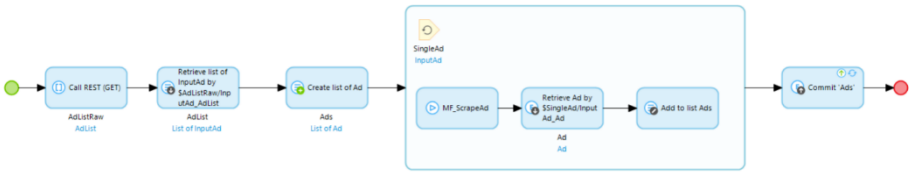

Fig. 4. Excerpt of the web crawler Mendix's microflows.

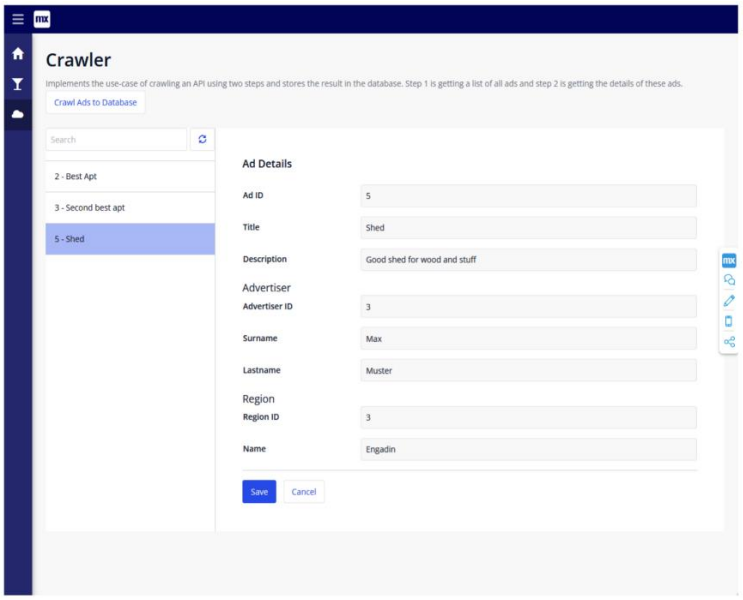

Fig. 5. Graphical user interface for managing the web crawler targets.

We asked the software developer to report on his perception while using Mendix to conduct the use case, including the design, the development process, and the improvement remarks. Based on the provided information, we concluded that Mendix allowed the software developer to implement a simplified web crawler based on the results. That means, although Mendix is a general-purpose MDD tool, the MDD tool provides enough functionalities to implement domain-specific software, as is a web crawler. Such a result is an insight for answering our RQ. However, we collected the software development improvement remarks about developing software using the MDD tool based on her/his experience. We analyzed such comments to compile them as opportunities in the following paragraphs.

Decreasing modeling complexity: The software developer stated that developing a simple feature requires several models, increasing the software development complexity. Based on his/her practical experience, the use case functionalities could be developed in a few lines of code (between 10 to 30 lines of code) on a general-purpose programming language such as JavaScript or C#. Therefore, we identified that providing MDD tools with general-purpose programming languages that allow software developers to write code and integrate them with the models can overcome such a complex issue.

Increasing MDD tool integrability: The software developer stated that several technologies and tools are usually integrated in practice to develop software such as web crawlers. Such integration allows software developers to increase development speed using tested and already-implemented functionalities. However, the software developer state that Mendix has no support for integrating domain-specific technologies and tools such as Puppeteer [17], a contemporary web crawling tool. Therefore, we identified that providing MDD tools with integration mechanisms can exploit the benefits of already-implemented technologies and tools, increasing the software development speed.

Improving modeling assistance: The software developer stated that he/she perceived some difficulties during modeling in Mendix. Finding references between models, debugging the microflows, and understanding the modeling syntax are examples of such problems. Although Mendix has modeling assistants to assist in creating models such as automatic completion, the software developer state that the modeling assistance in MDD tools is behind programming assistance in IDEs (Integrated Development Environment). This lack of well-designed and complete modeling assistance negatively affects the "developer" experience with the MDD tool. Therefore, we identify that improving modeling assistance in MDD tools by creating more complete and user-oriented modeling assistance can overcome such difficulties.

5 Conclusions and future work

Several MDD tools are available in the market, promising that software development teams can fast and easily develop any software by using them. However, what is the perception of a software developer in using an MDD tool to create software he/she is used to develop without models? We propose a use case in collaboration with a Swiss

company that allows users to compare insurances based on web crawling to address this question. We reviewed a set of available MDD tools and selected Mendix, a user-friendly and easy to configure MDD tool, to develop such a use case. Then, we ask an experienced software developer at the Swiss company to develop the use case. Although the software developer had no experience using MDD tools, we observed he/she successfully developed a simplified web crawler using Mendix. These results provide data for answering our RQ since, at least in this context, the selected MDD tool has enough functionalities to implement a web crawler. Finally, we collect the software developer remarks during the use case development using the MDD tool. As a result, we outlined three opportunities for improvement based on his/her experience: decreasing modeling complexity, increasing MDD tool interoperability, and improving modeling assistance.

Although this experience report's results are helpful, we know it is not feasible to generalize them based on just one software developer's perception, one specific domain, and one specific MDD tool. Thus, we plan to replicate this experience with other software developers, including domain-specific use cases in collaboration with other industrial partners. Currently, we have industry partners that can bring such domain-specific use cases to us, mainly focused on: software testing, data-centric applications, and car racing simulators. These efforts will bring us data for generalizing our results, arising new opportunities to improve the MDD tools.

Acknowledgments

Our research is supported by the Zürich University of Applied Sciences (ZHAW) – School of Engineering: Institute for Applied Information Technology (InIT); and the Innosuisse Flagship Initiative - Project SHIFT.

References

1. Sendall, S., Kozaczynski, W.: Model transformation: the heart and soul of model-driven software development. IEEE Software. 20, 42–45 (2003).
2. OutSystems Home Page, https://www.outsystems.com, last accessed 2022/03/04.
3. PowerApps Home Page, https://powerapps.microsoft.com/en-us/, last accessed 2022/03/04.
4. Appian Home Page, https://appian.com, last accessed 2022/03/04.
5. WebRato Home Page, https://www.webratio.com/site/content/es/home, last accessed 2022/03/04.
6. Mendix Home Page, https://www.mendix.com, last accessed 2022/03/04.
7. Jia, X., Jones, C.: AXIOM: A model-driven approach to cross-platform application development. ICSOFT 2012 - Proceedings of the 7th International Conference on Software Paradigm Trends. 24–33 (2012).
8. Acerbis, R., Bongio, A., Brambilla, M., Butti, S.: Model-Driven Development of Cross-Platform Mobile Applications with Web Ratio and IFML. Proceedings - 2nd ACM International Conference on Mobile Software Engineering and Systems, MOBILESoft 2015. 170–171 (2015).
9. Rieger, C.: Business apps with MAML. In: Proceedings of the Symposium on Applied Computing. pp. 1599–1606. ACM, New York, NY, USA (2017).

10.Rosales-Morales, V.Y., Sánchez-Morales, L.N., Alor-Hernández, G., Garcia-Alcaraz, J.L., Sánchez-Cervantes, J.L., Rodriguez-Mazahua, L.: ImagIngDev: A New Approach for Developing Automatic Cross-Platform Mobile Applications Using Image Processing Techniques. Computer Journal. 63, 732–757 (2020).
11.Khder, M.: Web Scraping or Web Crawling: State of Art, Techniques, Approaches and Application. International Journal of Advances in Soft Computing and its Applications. 13, 145–168 (2021).
12.Liddle, S.W.: Model-Driven Software Development. In: Handbook of Conceptual Modeling. pp. 17–54. Springer Berlin Heidelberg, Berlin, Heidelberg (2011).
13.Sahay, A., Indamutsa, A., di Ruscio, D., Pierantonio, A.: Supporting the understanding and comparison of low-code development platforms. In: 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). pp. 171–178. IEEE (2020).
14.Brambilla, M., Butti, S., Fraternali, P.: WebRatio BPM: A Tool for Designing and Deploying Business Processes on the Web. Presented at the (2010).
15.Geer, D.: Eclipse becomes the dominant Java IDE. Computer (Long Beach Calif). 38, 16–18 (2005).
16.Gartner, I.: Gartner Magic Quadrant for Enterprise Low-Code Application Platforms. (2021).
17.Puppeteer GitHub Repository, https://github.com/puppeteer/puppeteer, last accessed 2022/03/06.