Author pre-print. Paper accepted for EMMSAD 2023.
The final publication appeared in LNBIP. DOI: 10.1007/978-3-031-34241-7_20.

Opportunities in Robotic Process Automation by and for Model-Driven Software Engineering

Istvan David[1], Vasco Sousa[1], and Eugene Syriani[1]

DIRO, Université de Montréal, Canada
{firstname.lastname}@umontreal.ca

Abstract. Robotic Process Automation (RPA) offers a non-intrusive approach to workflow automation by defining and operationalizing automation rules through the graphical user interfaces of engineering and business tools. Thanks to its rapid development lifecycle, RPA has become a core enabler in many of nowadays' digital transformation efforts. In this paper, we briefly review how some of the critical success factors of RPA endeavors can be supported by the mature techniques of model-driven software engineering (MDSE); and how RPA can be used to improve the usability of MDSE tools. By that, we intend to shed light on the mutual benefits of RPA and MDSE and encourage researchers and practitioners to explore the synergies of the two fields. To organize such prospective efforts, we define a reference framework for integrating RPA and MDSE, and provide pointers to state-of-the-art RPA frameworks.

Keywords: RPA · MDSE · User Interfaces · Process automation · Automation

1 Introduction

Robotic Process Automation (RPA) aims to alleviate the human workload by automating workflows [1]. RPA offers a less intrusive alternative to traditional workflow automation: the integration of the automation technology is approached through the user interface of the software system. RPA bots emulate the user's behavior and interact with the information system through its user interface or by connecting to APIs to drive client servers, mainframes, or HTML code. Because the system subject to automation does not have to be altered, RPA offers a rapid development lifecycle and reduced development costs [2]. Often considered a core enabler of digital transformation with a high return on investment, RPA has been rapidly adopted by business-facing enterprises irrespective of their business sectors [3]. In recent years, numerous vendors have made substantial efforts to provide RPA platforms. Leading vendors like UiPath, Blue Prism, and Automation Anywhere offer enterprise-grade solutions in an integrated development and management platform. At the same time, large-scale business information system vendors, such as Microsoft, SAP, and IBM, have incorporated RPA into their portfolios by pivoting from traditional functionalities, e.g., BI and CRM [4].

Model-driven software engineering (MDSE) [5] is a paradigm in which software, including its processes (such as RPA processes), is modeled before it gets implemented. On the one hand, models provide means for formal analysis, something RPA processes could substantially benefit from. On the other hand, the success of MDSE heavily relies on the usability of engineering tools. The usability of MDSE tools, such as CAD tools and simulation software is far from efficient as complex engineering activities often rely on repetitive tasks [6]. RPA is a prime candidate to alleviate those issues. Despite these clear mutual benefits, synergies between RPA and MDSE have not been mapped yet. In this paper, we provide a brief overview of the relation between the two disciplines in both directions: RPA *by* and *for* MDE, and define a reference framework to organize future research and development on the topic.

State of the practice RPA frameworks. The latest (2022) Gartner Magic Quadrant report for RPA [7] lists four frameworks as clear market leaders. *UiPath*[1] offers features for governance, cloud-orchestrated RPA as a service, and intuitive UX for non-technical developers. Its main strength is the product strategy, centered around an integrated low-code platform. *Automation Anywhere*[2] is a leader in hyperautomation, i.e., combining RPA with machine learning (ML) capabilities. The Automation 360 platform offers RPA as a service, process discovery, analytics, and ML capabilities built on the TensorFlow framework that renders Automation Anywhere's screen capture accuracy superior to its competitors. *Microsoft Power Automate*[3] leverages the unparalleled ecosystem provided by the broader Microsoft Power Platform. Integration and orchestration capabilities with Power BI for analytics, Process Advisor for process mining, and Power Apps for low-code development are supported out-of-the-box. *Blue Prism*[4] offers cloud-based RPA solutions, governance tools, and advanced automation lifecycle management. With nearly 170,000 active RPA users, its customer ecosystem one of Blue Prism's strengths. Its industry-specific solutions and a wide array of enterprise application connectors make Blue Prism a leader in the RPA market.

2 Opportunities in RPA by MDSE

Recent in-depth surveys have identified key challenges and success factors of RPA endeavors [8,9]. Here, we outline three lines of research in MDSE (shown in Fig. 1) that could substantially contribute to the success of RPA.

2.1 Domain-specific languages for RPA configuration

As reported by Plattfaut et. al [9], ensuring sufficient process knowledge as the basis for automation and allowing the process owner to be part of the development are critical success factors of sound RPA solutions. However, current RPA

[1] `https://www.uipath.com/`

[2] `https://www.automationanywhere.com/`

[3] `https://powerautomate.microsoft.com/en-ca/robotic-process-automation`

[4] `https://www.blueprism.com/`

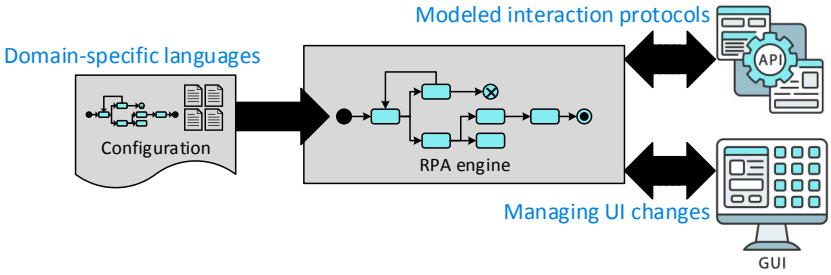

Fig. 1: Three potential contributions of MDSE to RPA

frameworks are shipped with mostly general-purpose process modeling languages that offer little-to-no customizability. This limits the involvement of process owners who typically possess highly domain-specific knowledge. In typical practical settings, however, RPA experts act as translators between the process owner and the RPA framework, and the success of capturing domain-specific workflows is entirely dependent on their understanding and interpretation, resulting in substantial accidental complexity. Domain-specific languages (DSLs) [10] could reduce this accidental complexity, narrow the cognitive gap between the domain expert and the RPA expert, and even allow expressing parts of the configuration, integration requirements, and test scenarios [9] by the domain expert. The modeling of multi-faceted RPA workflows might require an ensemble of modeling languages. While graphical notations for workflows are intuitive, their limitations are apparent when capturing complex structures. Multi-view Modeling [11] has been successfully employed in such scenarios and could provide foundations for modeling complex RPA configurations.

2.2 Explicitly modeled interaction protocols with APIs

Ensuring compliance with IT and security policies is another critical success factor in RPA [9]. Establishing supporting tools that reach across organization silos challenges the organizational embeddedness and audit of RPA processes.

One important technical facet is the interaction between the RPA process and the various IT systems across the organization. Such orchestration tasks are typically approached at the source code level that the RPA engine can use during the execution of the main workflow. Relying on source code to integrate APIs is problematic for two reasons. First, such an approach gives rise to unwanted accidental complexity. Inefficient source code can lead to the limited performance of the RPA workflow, reduced quality, and bugs. The lack of proper exception handling is a known shortcoming of RPA [8]. Second, certification of the RPA workflow might be of particular interest in settings when critical infrastructure is involved in the overall orchestration; or when the RPA workflow has to carry out tasks in a mission-critical manner (e.g., in the development of a critical cyber-physical system). Relying on source code substantially complicates any analysis, verification, and certification of the overall RPA configuration. Explicit

modeling of interaction protocols has been shown to be feasible and practical in developing complex engineering toolchains [12] and can translate well to RPA configurations. Statecharts and class diagrams have been suggested for modeling service interactions in engineering workflows [13], allowing the analysis and simulation of various behavioral properties, such as time-outs, exception handling, and parallelism. Finally, appropriately modeled interaction protocols enable the end-to-end simulation and optimization of the underlying workflow [14].

2.3 Managing changes to user interfaces

Flexibility and maintainability of RPA solutions, and support for their continuous evolution have been identified as additional success factors [8,9].

A severe shortcoming of RPA frameworks is their reliance on fixed user interfaces. Once an RPA configuration has been finalized, any change to the graphical user interface (GUI) might break it. This includes visible graphical elements that have been added, changed, or moved and HTML code not visualized on the GUI, such as changes to element identifiers or CSS classes. These limitations make RPA configurations brittle and could lead to substantial maintenance costs amidst unwanted vendor locking. Explicitly modeled GUI can significantly alleviate this problem [15]. Such approaches model the structure and behavior of the editor and generate the specific implementation from the models [16]. This enables associating RPA rules with the model itself instead of the specific GUI elements. Since models are manipulated through model transformations, the traceability of changes is explicit, enabling controlled evolution of RPA configurations based on the changes of the GUI.

3 Opportunities in RPA for MDSE

As much as RPA can benefit from MDSE, MDSE tools can benefit from RPA techniques as well. Complex modeling activities, such as developing model transformations and creating domain-specific languages, require repetitive tasks. Reusable and platform-specific workflow automation techniques, e.g., the Eclipse Modeling Workflow Environment, have been proposed to alleviate these problems. However, such approaches assume (i) the availability of advanced programming expertise and (ii) control over the business logic of the software tools at hand, e.g., open-source software. In such cases, the impact of RPA is moderate, as shown in Fig. 2. In the absence of programming skills and without control over the internals of tools, RPA becomes a viable and convenient alternative.

Here, we briefly discuss the primary and secondary candidates for RPA and provide example domains that rely heavily on model-driven techniques.

3.1 Primary candidates for RPA

MDSE settings in which non-programmer experts are working with proprietary tools are the primary candidates for using RPA. Examples of such settings include traditional non-software engineering disciplines, such as control software

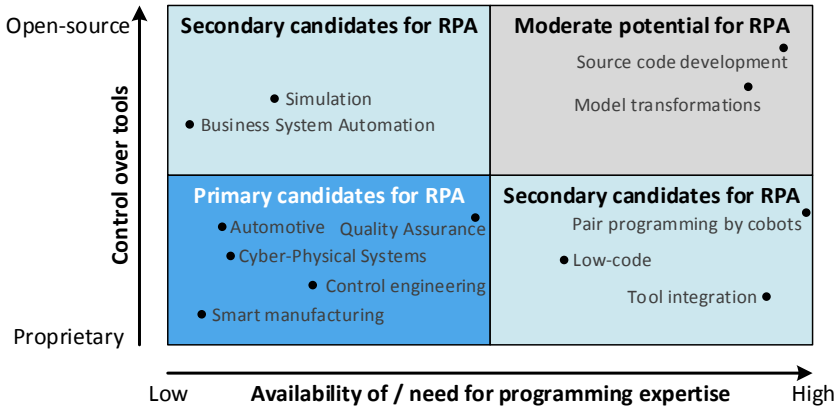

Fig. 2: Opportunities in using RPA for MDSE

and cyber-physical systems. While considered technical users, engineers in these domains do not necessarily possess the skills to automate their work using abstract process semantics. RPA can serve as a user-friendly alternative. Automotive and smart manufacturing engineering sectors are also reportedly lagging behind in digitalization [17], suggesting a high expected return on investment of digital improvement efforts, such as RPA. Screen capturing and workflow automation tools, such as Selenium [18], have been widely used for quality assurance purposes. However, these tools are limited to executing previously defined test cases on a graphical user interface. RPA-supported quality assurance opens up possibilities for automating the tester's tasks as well, including the selection, evaluation, and reporting of test cases, reconfiguring test cases, and improving test cases by incorporating historical data through machine learning.

3.2 Secondary candidates for RPA

Shifting from proprietary to open-source tools (top-left quadrant) allows for more control over the internals of the engineering tools, and enables automation through the business logic or back-end. Such settings limit the potential of RPA. However, the potential lack of available programming expertise might justify using RPA. Such cases can be observed in simulation disciplines where experts often rely on open-source tools and algorithms and business systems automation with non-technical users. Settings with proprietary tools but featuring advanced programming expertise (bottom-right quadrant) are another group of secondary candidates for RPA. RPA can successfully automate activities in developing low-code platforms with proprietary components and provide a workflow-centric orchestration functionality for lightweight tools integration. RPA can also emulate the human in pair programming settings, as advocated in eXtreme Programming and related disciplines.

4 Reference framework for integrating RPA and MDSE

We propose a conceptual reference framework to realize the opportunities in integrating RPA and MDSE in both directions. As depicted in Fig. 3, the framework lays the foundation for developing RPA solutions by and for MDSE. At a conceptual level, the framework relates these two directions between RPA and MDSE; and relates them to conventional software automation. The framework aims to aid RPA developers and tool providers to contextualize their efforts in terms of a typical socio-technological modeling and simulation system (STMSS), in which human and organizational factors must be considered during the system's lifecycle as well [19]. Furthermore, it aids MDSE developers and tool providers to incorporate RPA into their continuous development efforts, possibly in a hybrid approach combined with conventional software automation techniques.

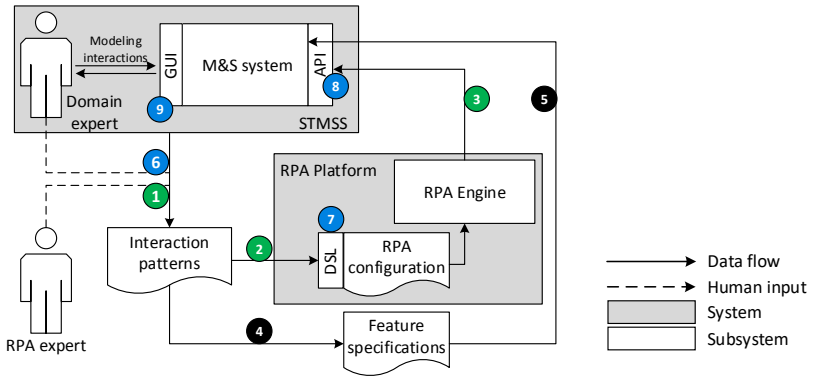

Fig. 3: Conceptual reference framework for integrating RPA and MDSE. ❶–❸ RPA for MDSE ❹–❺ Conventional Software Automation ❻–❾ RPA by MDSE

RPA for MDSE. RPA endeavors start with eliciting interaction patterns between the human modeler (i.e., the Domain expert) and the technical interfaces. Typically, the RPA expert performs this activity, indicated ❶ in Fig. 3. These interaction patterns are subsequently translated into the RPA configuration ❷. This transformation spans the process definitions, API interactions, file I/O operations, etc. Then, the RPA engine executes the process by interacting with the API of the modeling tool and other tools in the tool chain ❸.

Conventional Software Automation. In a traditional software automation approach, the elicited interaction patterns serve as the inputs to Feature specifications ❹. These feature specifications are subsequently implemented in the STMSS to provide better automation ❺. RPA offers a non-intrusive and faster development cycle for the automation of modeling and simulation tasks. In contrast, traditional software automation offers more robust solutions, thanks to the

tighter integration with systems in the tool chain. The two approaches are best used in combination. For example, RPA-based automation can serve as a temporary patch until developments of conventional software automation catch up with the changing requirements, or as the scaffolding for user-facing acceptance testing before conventional software engineering developments commence.

RPA by MDSE. Through MDSE techniques and tools, the Domain expert can directly express interaction patterns ❻. Such tools are, e.g., DSLs ❼, which provide high-level, concise mechanisms to specify RPA configurations. Both Domain experts and RPA experts can use syntax-directed editing and mixed textual-graphical editors for this purpose, possibly in a collaborative fashion [20]. State-of-the-art modeling and simulation systems often provide APIs for automation ❽, e.g., via scripting. RPA solutions might require more sophisticated API support, e.g., RESTful web APIs with more complex interaction protocols between the RPA engine and the STMSS while preserving scalability and portability. Finally, to address the adaptability challenges of RPA, modeling and simulation tool vendors might develop GUIs with explicitly modeled capabilities to provide information to the RPA infrastructure for adaptation purposes ❾.

5 Conclusion

In this paper, we have outlined the opportunities in employing MDSE for the improvement of RPA practices and employing RPA for improving MDSE tools. We outlined three lines of research for prospective MDSE researchers in support of improving the outlooks of RPA endeavors, especially in the areas of domain-specific languages, explicitly modeled interaction protocols, and modeled user interfaces. To assess opportunities in applying RPA in MDSE tools, we identified the openness of tools and the availability of programming expertise as the principal dimensions of MDSE settings that determine the potential impact of employing RPA. MDSE domains with proprietary tools or lacking advanced programming expertise are the primary candidates to leverage RPA. Such domains include automotive and CPS software engineering, various simulation domains, and low-code application development. Finally, we proposed a reference framework that illustrates how domain-specific RPA configuration languages, explicitly modeled API interactions, and generative techniques for building robust GUIs, are elements of the MDSE toolbox that can contribute to RPA immediately and improve its reliability and performance.

Future research should focus on the details of the outlined research directions, and on conducting surveys to validate and detail the matrix of opportunities. Adopters of RPA can use this paper as a guideline to better position their RPA efforts. Developers can use the pointers of this paper to make better choices when implementing new RPA features.

References

1. S. Aguirre and A. Rodriguez, "Automation of a business process using robotic process automation (RPA): A case study," in *Applied Computer Sciences in Engineering - 4th Workshop on Engineering Applications, WEA 2017, Colombia, September 27-29, 2017, Proceedings*, ser. CCIS, vol. 742. Springer, 2017, pp. 65–71.
2. A. Asatiani and E. Penttinen, "Turning robotic process automation into commercial success – case opuscapita," *Journal of Information Technology Teaching Cases*, vol. 6, no. 2, pp. 67–74, 2016.
3. J. Siderska, "Robotic process automation – A driver of digital transformation?" *Engineering Management in Production & Services*, vol. 12, no. 2, pp. 21–31, 2020.
4. B. Schaffrik, "The Forrester Wave™: Robotic Process Automation, Q1 2021 – The 14 Providers That Matter Most And How They Stack Up," 2021.
5. M. Brambilla, J. Cabot, and M. Wimmer, *Model-Driven Software Engineering in Practice, Second Edition*. Morgan & Claypool Publishers, 2017.
6. M. Gamboa and E. Syriani, "Improving user productivity in modeling tools by explicitly modeling workflows," *Softw. Syst. Model.*, vol. 18, pp. 2441–2463, 2019.
7. S. Ray *et al.*, "Magic quadrant for robotic process automation," *Von Gartner: https://www. gartner. com/doc/reprints*, 2021.
8. R. Syed *et al.*, "Robotic process automation: Contemporary themes and challenges," *Comput. Ind.*, vol. 115, p. 103162, 2020.
9. R. Plattfaut *et al.*, "The critical success factors for robotic process automation," *Comput. Ind.*, vol. 138, p. 103646, 2022.
10. M. Fowler, *Domain-Specific Languages*, ser. The Addison-Wesley signature series. Addison-Wesley, 2011.
11. M. Verlage, "Multi-view modeling of software processes," in *European Workshop on Software Process Technology*. Springer, 1994, pp. 123–126.
12. M. Biehl, J. El-khoury, F. Loiret, and M. Törngren, "On the modeling and generation of service-oriented tool chains," *Softw. Syst. Model.*, vol. 13, no. 2, pp. 461–480, 2014.
13. S. Van Mierlo *et al.*, "A multi-paradigm approach for modelling service interactions in model-driven engineering processes," in *Proceedings of the Model-driven Approaches for Simulation Engineering Symposium, SpringSim 2018, Baltimore, MD, USA, April 15-18, 2018*. ACM, 2018, pp. 6:1–6:12.
14. I. David, H. Vangheluwe, and Y. Van Tendeloo, "Translating engineering workflow models to DEVS for performance evaluation," in *2018 Winter Simulation Conference, WSC 2018, Sweden, December 9-12, 2018*. IEEE, 2018, pp. 616–627.
15. V. Sousa, E. Syriani, and K. Fall, "Operationalizing the integration of user interaction specifications in the synthesis of modeling editors," in *Proceedings of the 12th ACM SIGPLAN International Conference on Software Language Engineering, SLE 2019, Athens, Greece, October 20-22, 2019*. ACM, 2019, pp. 42–54.
16. E. Syriani, D. Riegelhaupt, B. Barroca, and I. David, "Generation of custom textual model editors," *Modelling*, vol. 2, no. 4, pp. 609–625, 2021.
17. J. Bughin, J. Manyika, and T. Catlin, "Twenty-five years of digitization: Ten insights into how to play it right," *Boston: McKinsey Global Institute*, 2019.
18. A. Bruns, A. Kornstädt, and D. Wichmann, "Web application tests with selenium," *IEEE Softw.*, vol. 26, no. 5, pp. 88–91, 2009.
19. G. D. Baxter and I. Sommerville, "Socio-technical systems: From design methods to systems engineering," *Interact. Comput.*, vol. 23, no. 1, pp. 4–17, 2011.
20. I. David and E. Syriani, "Real-time Collaborative Multi-Level Modeling by Conflict-Free Replicated Data Types," *Software & Systems Modeling*, 2022.