A Tutorial on Software Engineering Intelligence: Case Studies on Model-Driven Engineering

Marouane Kessentini
Department of Computer Science
University of Michigan-Dearborn
Michigan, USA
marouane@umich.edu

Xin Yao
Department of Computer Science
SUSTech
Shenzhen, China
xiny@sustc.edu.cn

Kalyanmoy Deb
Department of Electrical and Computer Engineering
Michigan State University
Michigan, USA
kdeb@egr.msu.edu

Abstract—**The recent advances in Artificial Intelligence (AI) are dramatically impacting the way we are modelling software systems. A large number of computational intelligence based approaches and tools, combining computational search and machine learning, proved to be successful in automating and semi-automating several activities to support developers. However, the adoption of computational intelligence to address model driven engineering problems is still under-explored. In this tutorial, we will give an overview about computational intelligence, why model-driven engineering is a suitable paradigm for computational intelligence and how computational intelligence could benefit from the recent advances in model-driven engineering. Then, we will focus on some case studies that we published around the adaptation of a variety of computational intelligence techniques for model transformations, models evolution, model changes detection, co-evolution, model/metamodel refactoring, models merging, models quality and metamodels matching. To make the tutorial interactive, the participants will have the opportunity to practice our interactive intelligent MDE tools during the tutorial. Finally, we will conclude the tutorial with different suggestions to enhance the adoption of MDE intelligence research into industry, and the lessons that we learned along this journey and our vision about the future of MDE intelligence. The event will target a wide range of researchers and practitioners from both the model-driven engineering and computational intelligence communities and will reduce the gap between them. The participants will learn about the recent advances in computational intelligence and acquire the required skills to apply them for relevant MDE problems.**

Model-Driven Engineering, computational intelligence, model transformations, models refactoring, machine learning

I. Short Bio of the Presenters

The three organizers of the tutorial have extensive experience and complementary expertise on both MDE and Computational Intelligence. Dr. Kessentini is the PC chair of the foundations track of MODELS2019, he will be the general chair of ASE2021 and he organized with Dr. Deb the search based software engineering symposium (SSBSE2016).

Dr. Marouane Kessentini is a recipient of the prestigious 2018 President of Tunisia distinguished research award, the University of Michigan distinguished teaching award, the University of Michigan distinguished digital education award, the Collge of Engineering and Computer Science distinguished research award, 4 best paper awards, and his AI-based software refactoring invention, licensed and deployed by industrial partners, is selected as one of the Top 8 inventions at the University of Michigan for 2018, among over 500 inventions, by the UM Technology Transfer Office. Prior to joining UM in 2013, He received his Ph.D. from the University of Montreal in Canada in 2012. He received several grants from both industry and federal agencies and published over 110 papers in top journals and conferences. Dr. Kessentini has several collaborations with industry on the use of computational search, machine learning and evolutionary algorithms to address software engineering and services computing problems. He is the co-founder of IWoR and NasBASE, General Chair of SSBSE16 and ASE21, and PC chair of MODELS19, GECCO14-15. He served as invited speaker at SSBSE and WCCI, graduated 13 Ph.D. student as a chair and serving as associate editor in 7 journals and PC member of over 100 conferences. He organized tutorials on Search based Software Engineering at SSBSE2018, WCCI2016, ASE2016, etc. He has extensive publications on adopting computational intelligence in MDE [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13] and organized several workshops at MODELS (2013, 2014 and 2015) around this topic.

Xin Yao is a Chair Professor of Computer Science at the Southern University of Science and Technology, Shenzhen, China, and a part-time professor at the University of Birmingham, UK. His major research interests include evolutionary computation, ensemble learning and search-based software engineering. His work won the 2001 IEEE Donald G. Fink Prize Paper Award, 2010, 2015 and 2017 IEEE Transactions on Evolutionary Computation Outstanding Paper Awards, 2010 BT Gordon Radley Award for Best Author of Innovation (Finalist), 2011 IEEE Transactions on Neural Networks Outstanding Paper Award, and many other best paper awards. He received the prestigious Royal Society Wolfson Research Merit Award in 2012 and the IEEE CIS Evolutionary Computation Pioneer Award in 2013. He has extensive publications on search based software engineering.

Kalyanmoy Deb is Koenig Endowed Chair Professor at Department of Electrical and Computer Engineering in Michigan State University, USA. Prof. Deb's research interests are in evolutionary optimization and their application in multi-criterion optimization, modeling, and machine learning. He was awarded IEEE EC Pioneer award, Infosys Prize, TWAS Prize in Engineering Sciences, CajAstur Mamdani Prize, Distinguished Alumni Award from IIT Kharagpur, Edgeworth-Pareto award, Bhatnagar Prize in Engineering Sciences, and Bessel Research award from Germany. He is fellow of IEEE and ASME. He has published over 500 research papers with Google Scholar citation of over 122,000 with h-index 112. He is in the editorial board on 18 major international journals.

II. Proposed Length

A half-day (3 hours).

III. The Proposed Tutorial

More effective software development has the potential to make the infrastructure on which so many aspects of our society depend less costly including banks, businesses and government agencies. The increasing size and complexity of software systems make current development processes expensive, laborious and error prone. This evolution of software systems currently requires enormous human effort, forcing highly skilled engineers to waste significant time adapting many tedious requirements, planning, design, modeling, implementation, quality and testing details. Thus, it is becoming more challenging for humans to design and maintain software without automated and semi-automated tools to support them in their tasks.

The recent advances in Artificial Intelligence (AI) are dramatically impacting the way we are building and modeling software systems. A large number of AI based approaches and tools, such as search based software engineering (SBSE), proved to be successful in automating and semi-automating several activities to support developers from initial requirements, design and modeling, project planning, documentation generation and cost estimation to regression testing, debugging and evolution. There is also an increasing interest in AI based tools from the industrial sector such as work on testing involving Google and Microsoft, work on requirements analysis involving Motorola and NASA, Kessentini and Deb work on design refactoring involving Ford, eBay, Philips and SEMA, and work on automated documentation and interactive modeling with ABB.

Software Engineering Intelligence is a fundamental discipline which impacts a wide spectrum of software engineering activities. Research advances in this area will have a significant impact on Computational Intelligence, Natural Language Processing and other related areas. We identified over 1,879 papers and 2,684 authors from various disciplines and countries working on related Software Engineering Intelligence topics. Model-Driven Engineering is one of the most important topics where computational intelligence techniques has been applied with around 392 papers mainly based on evolutionary algorithms and mining techniques.

Considering the cross-cutting nature of AI based MDE and the influence it exerts on other areas of MDE, machine learning, natural language processing and computational intelligence, a wide range of research opportunities will be enabled by this tutorial. We are confident that the development of this community will increase the rigor of both MDE and computational intelligence research and enable opportunities which are currently missed. Several researchers in MDE are lacking the required background in computational intelligence and vice-versa. Thus, this tutorial will reduce the existing gab between the two communities and encourage the adoption of computational intelligence in MDE.

It is difficult today to: (1) combine different types of AI algorithms ranging from metaheuristics search to machine learning and deep learning for MDE problems due to the different assumptions that these tools make (e.g., the execution environments, formats used, modeling languages, etc.) which prevent their inhibiting breakthroughs. However, the majority of recent work in AI based MDE show that MDE problems should be addressed using a combination of AI techniques such as NLP and machine learning or metaheuristics and clustering algorithms; (2) transfer results of successful research to industry and enable easy entry for new researchers into the field due to the required expertise in both MDE and AI. Furthermore, practitioners still face a major difficulty to find, understand, use and configure AI algorithms for MDE problems such as design refactoring and model transformations. Most of researchers lack the resources, including data, needed to build tools that are robust and scalable enough to be easily and effectively evaluated and used by practitioners in an industrial setting. Furthermore, the validation of AI based approaches for MDE is challenging due to the randomness involved in these algorithms thus researchers need a clear guide on how to statistically and scientifically validate them; (3) access AI based tools for MDE since they are unavailable, defective, or are no longer supported by their original authors. For tools that do work, it is common for them to not operate as expected due to the challenging parameters tuning of AI tools resulting in major effort required to adapt these tools for further research and the difficulty to reproduce existing results; (4) find Benchmarks and Datasets. Scalability is one of the main motivations to use AI techniques but it is hard to evaluate using existing small datasets. Currently most of the new computational intelligence techniques are first validated on artificial benchmarks (e.g., the ZDT and DTLZ benchmarks). Similarly, most NLP-based techniques in MDE are first validated on datasets that were prepared by the authors of the techniques. AI based MDE community is in great need of benchmarks and test datasets for common MDE tasks, such as model transformations; (5) adapt and test novel AI techniques by MDE researchers and practitioners due to the absence of an infrastructure that can offer a template to follow in order to adapt an AI technique for a specific MDE problem without the need for strong expertise in AI.

The main goal of this tutorial is to give MDE researchers a clear understanding of the recent advances in computational intelligence, why MDE is relevant for computational intelligence, how to adapt a computational intelligence technique to an MDE problem illustrated with many real-world example and tools to practice and discuss the future of MDE intelligence.

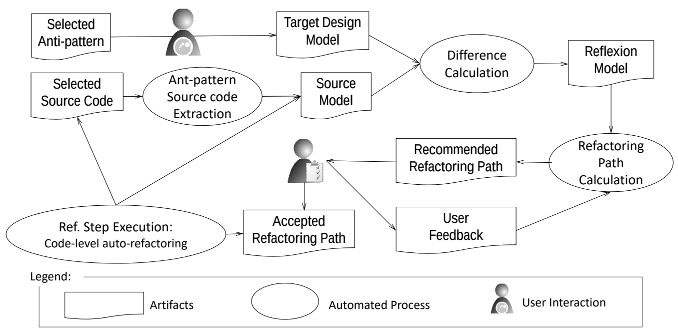

Fig. 1. The Interactive Design Refactoring Process

A. An Example of a Case Study on Design Refactoring

Although low-level code quality issues can largely be automated, higher-level refactoring—such as redistributing functionality into different components, decoupling a large code base into smaller modules, redesigning to a design pattern,—requires abstractions determined by the developers of the system. In these cases, these developers usually has a desired design in mind as the refactoring target, and they needs to conduct a series of low-level code changes to achieve this target. Without explicit guidance about which path to take, such restructuring tasks can be demanding for scientists with limited knowledge of best design practices. One of our industrial partners took several weeks to refactor the architecture of a medium-size project [14]. Tokuda and Batory [15] presented two case studies where architectural refactoring involved more than 800 steps, estimated to take more than 2 weeks.

Given that fully automatic refactoring usually does not lead to the desired architecture and that a designer's feedback should be included, we proposed *an interactive architecture refactoring recommendation system* to integrate higher-level abstractions from humans with lower-level refactoring automation.

We proposed an interactive design refactoring support framework as illustrated in Figure 1. The process starts from the selection of one or more quality issues to address by the user based on the outputs of the two previous detection and prioritization steps, and the associated source code. From the code, we can reverse-engineer a high-level model, such as a UML class diagram, of these quality issues [1], [2], then apply the following major steps:

1. Supporting target design modeling. Target Design Models provided by human designers will express key abstractions that cannot be automatically obtained. In our earlier work [16], [1], [2], users could draw a high-level model as the target design and the differences between the source model and the target design form a Reflexion Model that was visualized. Our work also revealed the challenges of asking practitioners to provide a high-level target model especially with little background in the use of best design practices. In this tutorial, we will explore how to allow the user to express high-level designs without interfering in their development process or requiring extensive background in software architecture, model-driven engineering and quality, and calculate a Reflexion Model to reveal discrepancies between them. Next we will explain the guidance that we can offer to designers to find different possible design alternatives of their systems without the need to manually specify them.

2. Refactoring path recommendation. Our framework can calculate optimal refactoring "paths", using multi-objective intelligent algorithms with conflicting quality attributes that can be used to evaluate a new design. Given a reflexion model, we will calculate a set of paths, each consisting of an ordered list of atomic code-level refactoring steps. We formulated multi-objective search-based formulation to find and recommend an optimal refactoring path based on the following criteria: 1) maximizing the consistencies between the target and source; 2) improving OO design quality, such as coupling and cohesion; and 3) maximizing the similarity between the target and source design. Multi-objective search enables us to find the best trade-offs in avoiding interference between multiple refactoring steps and still reach a target design, as closely as possible without the need to have the designers manually defining the targeted design.

3. Refactoring execution with user feedback. Given a recommended refactoring path, the user can examine each step and decide whether to accept, reject, or ignore it. Our tool will then execute the accepted steps to update the Source Model, and then calculate the updated differences and begin a new iteration. The designer's decisions about accepting/rejecting recommendations are recorded as User Feedback, which will be considered when computing the next round of recommendations. The refactoring process ends when no discrepancies exist in the reflexion model, or the tool cannot produce further refactoring recommendations to eliminate the discrepancies. We were able to reduce the time of refactoring from several weeks (conduced by our collaborators manually) to a few hours, demonstrating the feasibility and potential of the proposed approach, as well as the need for more sophisticated, less intrusive support for integrating human feedback [14], [17], [18], [19], [20], [21], [22]. We believe that the integration of this new interaction process with the users who are non-programming experts, but mainly designers and architects, will make quality issues easier to understand and address.

The main novelty of the tutorial can be summarized as following:

- An overview about the recent advances in computational intelligence and how they could be useful to address relevant MDE problems. We have few opportunities in the MDE community for connections with leading researchers in Computational intelligence.
- Practicing a set of tools related to published papers by the presenters on addressing MDE using computational intelligence. Some of these tools are licensed to industrial partners of Dr. Kessentini.
- Discussions around how MDE could be relevant for computational intelligence to design explainable techniques.

References

[1] M. W. Mkaouer, M. Kessentini, S. Bechikh, K. Deb, and M. Ó Cinnéide, "Recommendation system for software refactoring using innovization and interactive dynamic optimization," in *Proceedings of the International Conference on Automated Software Engineering*, 2014, pp. 331–336.

[2] A. Ouni, M. Kessentini, H. A. Sahraoui, K. Inoue, and K. Deb, "Multi-criteria code refactoring using search-based software engineering: An industrial case study," *ACM Trans. Softw. Eng. Methodol.*, vol. 25, no. 3, pp. 23:1–23:53, 2016. [Online]. Available: https://doi.org/10.1145/2932631

[3] M. Kessentini, H. A. Sahraoui, and M. Boukadoum, "Model transformation as an optimization problem," in *Model Driven Engineering Languages and Systems, 11th International Conference, MoDELS 2008, Toulouse, France, September 28 - October 3, 2008. Proceedings*, 2008, pp. 159–173. [Online]. Available: https://doi.org/10.1007/978-3-540-87875-9_12

[4] A. Ghannem, M. Kessentini, M. S. Hamdi, and G. El-Boussaidi, "Model refactoring by example: A multi-objective search based software engineering approach," *Journal of Software: Evolution and Process*, vol. 30, no. 4, 2018. [Online]. Available: https://doi.org/10.1002/smr.1916

[5] M. Kessentini, U. Mansoor, M. Wimmer, A. Ouni, and K. Deb, "Search-based detection of model level changes," *Empirical Software Engineering*, vol. 22, no. 2, pp. 670–715, 2017. [Online]. Available: https://doi.org/10.1007/s10664-016-9442-8

[6] M. Fleck, J. Troya, M. Kessentini, M. Wimmer, and B. Alkhazi, "Model transformation modularization as a many-objective optimization problem," *IEEE Trans. Software Eng.*, vol. 43, no. 11, pp. 1009–1032, 2017. [Online]. Available: https://doi.org/10.1109/TSE.2017.2654255

[7] A. Ghannem, G. El-Boussaidi, and M. Kessentini, "On the use of design defect examples to detect model refactoring opportunities," *Software Quality Journal*, vol. 24, no. 4, pp. 947–965, 2016. [Online]. Available: https://doi.org/10.1007/s11219-015-9271-9

[8] B. Alkhazi, T. Ruas, M. Kessentini, M. Wimmer, and W. I. Grosky, "Automated refactoring of ATL model transformations: a search-based approach," in *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems, Saint-Malo, France, October 2-7, 2016*, 2016, pp. 295–304. [Online]. Available: http://dl.acm.org/citation.cfm?id=2976782

[9] U. Mansoor, M. Kessentini, P. Langer, M. Wimmer, S. Bechikh, and K. Deb, "MOMM: multi-objective model merging," *Journal of Systems and Software*, vol. 103, pp. 423–439, 2015. [Online]. Available: https://doi.org/10.1016/j.jss.2014.11.043

[16] V. Alizadeh and M. Kessentini, "Reducing interactive refactoring effort via clustering-based multi-objective search," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineer-*

[10] D. Sahin, M. Kessentini, M. Wimmer, and K. Deb, "Model transformation testing: a bi-level search-based software engineering approach," *Journal of Software: Evolution and Process*, vol. 27, no. 11, pp. 821–837, 2015. [Online]. Available: https://doi.org/10.1002/smr.1735

[11] M. W. Mkaouer and M. Kessentini, "Model transformation using multiobjective optimization," *Advances in Computers*, vol. 92, pp. 161–202, 2014. [Online]. Available: https://doi.org/10.1016/B978-0-12-420232-0.00004-0

[12] A. ben Fadhel, M. Kessentini, P. Langer, and M. Wimmer, "Search-based detection of high-level model changes," in *28th IEEE International Conference on Software Maintenance, ICSM 2012, Trento, Italy, September 23-28, 2012*, 2012, pp. 212–221. [Online]. Available: https://doi.org/10.1109/ICSM.2012.6405274

[13] M. Kessentini, H. A. Sahraoui, M. Boukadoum, and O. Benomar, "Search-based model transformation by example," *Software and System Modeling*, vol. 11, no. 2, pp. 209–226, 2012. [Online]. Available: https://doi.org/10.1007/s10270-010-0175-7

[14] V. Alizadeh, M. Kessentini, W. Mkaouer, M. Ocinneide, A. Ouni, and Y. Cai, "An interactive and dynamic search-based approach to software refactoring recommendations," *IEEE Transactions on Software Engineering*, 2018.

[15] L. Tokuda and D. Batory, "Evolving object-oriented designs with refactorings," in *Proceedings of International Conference on Automated Software Engineering*, 1999, pp. 174–181.

ing, ASE 2018, Montpellier, France, September 3-7, 2018, 2018, pp. 464–474. [Online]. Available: https://doi.org/10.1145/3238147.3238217

[17] M. Fleck, J. Troya, M. Kessentini, M. Wimmer, and B. Alkhazi, "Model transformation modularization as a many-objective optimization problem," *IEEE Transactions on Software Engineering*, 2017.

[18] A. Ouni, R. G. Kula, M. Kessentini, T. Ishio, D. M. German, and K. Inoue, "Search-based software library recommendation using multi-objective optimization," *Information and Software Technology*, vol. 83, pp. 55–75, 2017.

[19] H. Wang, M. Kessentini, and A. Ouni, "Bi-level identification of web service defects," in *International Conference on Service-Oriented Computing*. Springer, 2016, pp. 352–368.

[20] M. W. Mkaouer, M. Kessentini, M. Ó. Cinnéide, S. Hayashi, and K. Deb, "A robust multi-objective approach to balance severity and importance of refactoring opportunities," *Empirical Software Engineering*, vol. 22, no. 2, pp. 894–927, 2017.

[21] U. Mansoor, M. Kessentini, M. Wimmer, and K. Deb, "Multi-view refactoring of class and activity diagrams using a multi-objective evolutionary algorithm," *Software Quality Journal*, vol. 25, no. 2, pp. 473–501, 2017.

[22] M. Kessentini, R. Mahaouachi, and K. Ghedira, "What you like in design use to correct bad-smells," *Software Quality Journal*, vol. 21, no. 4, pp. 551–571, 2013.