Cognifying Model-Driven Software Engineering

Jordi Cabot[1,2], Robert Clarisó[2], Marco Brambilla[3], Sébastien Gérard[4]

[1]ICREA, [2] Universitat Oberta de Catalunya, [3]Politecnico di Milano, [4]CEA List
jordi.cabot@icrea.cat, rclariso@uoc.edu,
marco.brambilla@polimi.it, sebastien.gerard@cea.fr

Abstract. The limited adoption of Model-Driven Software Engineering (MDSE) is due to a variety of social and technical factors, which can be summarized in one: its (real or perceived) benefits do not outweigh its costs. In this vision paper we argue that the *cognification* of MDSE has the potential to reverse this situation. Cognification is the application of knowledge (inferred from large volumes of information, artificial intelligence or collective intelligence) to boost the performance and impact of a process. We discuss the opportunities and challenges of cognifying MDSE tasks and we describe some potential scenarios where cognification can bring quantifiable and perceivable advantages. And conversely, we also discuss how MDSE techniques themselves can help in the improvement of AI, Machine learning, bot generation and other cognification techniques

Keywords: model, machine learning, bot, model-driven, AI

1 Introduction

It is hard to imagine anything that would "change everything" as much as cheap, powerful, ubiquitous intelligence and exploitation of knowledge. Even a very tiny amount of useful intelligence embedded into an existing process boosts its effectiveness to a whole other level [18]. This process is known as *cognification*.

Cognification can be defined as the application of knowledge to boost the performance and impact of a process. It does not restrict itself to what we typically refer to as artificial intelligence, with Deep Learning as its latest and hottest technique. Cognification includes as well the combination of past and current human intelligence (i.e. all current humans online and the actions they do). Under this definition, collective intelligence and crowdsourcing [29] are valid cognification tools. Cognification does not imply either the existence of a super AI but the availability of highly specialized intelligences that can be plugged in depending on the needs of the problem to be solved.

Several initiatives aim to cognify specific tasks in Software Engineering, for instance, using machine learning (ML) for requirements prioritization [25], for estimating the development effort of a software system [30] or the productivity of individual practitioners [22] or to predict defect-prone software components [26]. This trend is also happening at the tool level, e.g., the recent Kite IDE[1] claims to "augment your

[1] https://kite.com/

coding environment with all the web's programming knowledge". MDSE should join this ongoing trend.

Moreover, we know the limited adoption of MDSE is due to a variety of social and technical factors [14] but we can summarize them all in one: its benefits do not outweigh its costs. We believe cognification could drastically improve the benefits and reduce the costs of adopting MDSE.

In this paper we discuss the opportunities that cognification can bring to the MDSE field, in terms of possible tools, process steps, and usage scenarios and their empowerment through cognification. We also discuss how MDSE itself can be applied to AI and knowledge–aware technology development, and we conclude with some challenges and risks that this roadmap may encompass.

2 Opportunities in the Cognification of MDSE

All MDSE tasks and tools are good candidates to be cognified, since they typically involve plenty of knowledge–intensive activities. Here, we comment on a few examples where the performance or quality of experience can be especially boosted thanks to cognification.

2.1 Modeling Bots

Cognification can enable **modeling bots** playing the role of virtual modeling assistant. Based on previous models on the same domain available online or on a comparison between your current model and general ontologies, the bot could suggest missing properties on the model, recommend best practices or warn that some model aspects differ from the way they are typically modeled.

Preliminary work in this direction considers recommendations during the construction of a model to reduce the effort made by designers. [28] proposes a recommender that suggests which model element should be the target of a newly created reference. Similarly, [21] studies the notion of an *"auto-complete"* that is able to infer the intended pattern that is being designed.

In some contexts, models are created by domain experts by using a Domain-Specific Language (DSL). This creation process includes manual activities like auto-completing a partial model or introducing small fixes to ensure it satisfies the integrity constraints of the domain. *Proactive modeling* [24] aims to automate these manual activities. Bots could even interact with domain experts in a more friendly environment (e.g. social networks)[2] to facilitate the knowledge acquisition process.

2.2 Model Inferencers

A **model inferencer** is able to deduce a common schema behind a set of unstructured data (logs, tweets, forum messages,...). This model would be a useful "lense" to interpret the data. A good example of a model inferences is JSON Discoverer [17]. Given a

[2] https://saraperezsoler.github.io/ModellingBot/

(set of) JSON documents, it analyzes the JSON data and generates for you a class diagram showing graphically the implicit JSON schema of your documents plus an object diagram representing their data

In a related context, *process mining* [1] aims to achieve a similar goal with respect to procedural information describing a process, action or activity.

2.3 Smart Code Generators

A cognified **code generator** would be able to learn the style and best practices of a particular organization and mimic them in the code it outputs. By learning from good code samples, the generator would be able to imitate the company's best practices and style and maximize its chances to be accepted as a new "developer" for the company.

This line of thought is close to existing works on example-based generation of modeling artefacts. For instance, [19], where a search-based approach is used to to derive model-to-model transformations from sample input and output models.

2.4 Real-time Model Reviewers

Machine learning has already been used in the context of verification and validation to tune verification algorithms by selecting the best choice for heuristics [15, 12]. However, here the aim would be constructing a **real-time model reviewer** able to give continuous feedback on the quality of the model from the point of view of consistency and correctness.

Such tool would resort to different verification and validation (V&V) techniques, using information about previous analysis to predict the complexity of the analysis [16] and select the most suitable method to verify a particular model. This problem is known in the A.I. field as the *algorithm selection problem* [20].

In addition to selecting the most adequate tool, previous experiences can be useful by highlighting the most frequent faults or the constructs that are most likely to produce problems. This type of information can be used to guide the search process, e.g. which kind of properties should be checked for a particular model. Some preliminary works addressed the use of semantic reasoning for validation and property checking for models: for instance [9], propose a simple tool able to address issues that cannot be identified by traditional type checking tools.

Another approach to model quality would be to *evolve* or *adapt* models rather than detect and correct faults. For instance, the notion of *liquid models* [23], which considers that design-time models should not be immutable artifacts, but that they should evolve to take into account potential deviations occurring at run-time. Cognification could be used as a means to study and explore candidate models during this evolution process.

2.5 Advanced Self-Morphing and Collaborative Modeling Tools

Automatic learning approaches can also be used within **self-morphing modeling tool**, able to adapt its interface, functions, behaviour (and even the expressiveness of the language offered) to the expertise of the tool user and to the context of the modeling problem addressed. This challenge is related to the problem of plastic user interfaces, i.e. interfaces able to keep its usability under changing uses and circunstances (responsive websites can be regarded as a limited example of a plastic UI). While there has been some work on using MDE to generate plastic UIs [27], building a plastic MDE IDE remains an open challenge.

The tool should not just adapt to different user profiles but also effectively support the collaboration of those users. Indeed, congnification also comprises exploitation of human and collective intelligence. As an example, we could leverage crowdsourcing techniques [6] to devise the best domain-specific modeling language to be used by the tool in order to optimize the communication process between modeling experts and domain experts. This has been so far applied to the the problem of agreeing on the concrete syntax notation of domain-specific languages, as described in [4].

2.6 Semantic Reasoning Platforms, Explainability and Storification

At the purpose of making models as self-explanatory as possible, semantics-based techniques can be applied to the concepts in the model to enable automatic explanation of models. This would require a **semantic reasoning platform** able to map modeled concepts to existing ontologies and provide definitions (similar to Kindle WordWise[3]), references, relations to relevant concepts, services and contextual information; and conversely also generate new knowledge through inference.

This would also enable automatic generation of thesaurus, data dictionaries, and even explanatory text out of models. Integrating MDE with resources such as ontologies, semantic processors, NLP tools, rule–based inference engines, and alike will be crucial in this setting.

2.7 Scalable Model and Data Fusion Engine

Big data architectures and data streaming platform enable the construction of **data fusion engines** that are able to perform semantic integration and impact analysis of design-time models with runtime data such as software usage logs, user profiles and so on. A typical use case is the integration of Web application logs, which are widespread in Web servers, with user interaction models of Web applications. An example is the work [2, 3] which integrates with a scalable approach the real-time big data stream coming from large–scale Web sites with IFML models[7]. This close interaction between design and runtime models is also the focus of the MegaMart2 EU project [4].

3 Model-driven Engineering of AI and Knowledge-aware Software

Besides the advantages that cognification can bring to MDSE, the reverse is also true: MDSE techniques can be used to improve knowledge-aware technologies. As in any other domain, the use of models (and its abstraction power) can help simplify, interoperate and unify a fragmented technological space, as it is the case right now in AI, with

[3] https://www.amazon.com/gp/feature.html?ie=UTF8&docId=1002989731

[4] https://megamart2-ecsel.eu/

plenty of competing and partially overlapping libraries and components for all kinds of knowledge processing tasks.

So far, this line of work remains largely unexplored. A few exceptions are [13], aiming to integrate machine learning results in domain modeling, [10], aligning MDE and ontologies, and works oriented to model-driven development of semantic Web based applications [8] and semantic web services [5].

4 Conclusions and challenges

As we described in this paper, a lot of possible scenarios and tools in MDSE can benefit from the application of cognification techniques in broad sense. Even if only a few of them become a reality in the short-term, they have the potential to drastically change the way MDSE is used and perceived in the software community.

Nevertheless, some risks need to be addressed. The main challenge is that most of the above scenarios imply the availability of lots of training data in order to get high-quality learning results. In MDSE, this means a *curated catalog* of good and bad models, meta-models, transformations, code, and so on [11]. While some model repositories are available (e.g. REMODD [5] or MDEForge [6] the number and diversity of the modeling artefats they contain is still limited. Improving this situation is a strong requirement towards the cognification of MDE.

References

1. van der Aalst, W., et al: Process Mining Manifesto, pp. 169–194. Springer (2012)
2. Bernaschina, C., Brambilla, M., Koka, T., Mauri, A., Umuhoza, E.: Integrating modeling languages and web logs for enhanced user behavior analytics. In: WWW'17. pp. 171–175 (2017)
3. Bernaschina, C., Brambilla, M., Mauri, A., Umuhoza, E.: A big data analysis framework for model-based web user behavior analytics. In: Web Engineering - 17th International Conference, ICWE 2017, Rome. pp. 98–114 (2017)
4. Brambilla, M., Cabot, J., Izquierdo, J.L.C., Mauri, A.: Better call the crowd: Using crowdsourcing to shape the notation of domain-specific languages. In: Proc. of 2017 ACM SIGPLAN International Conference on Software Language Engineering (SLE?17). ACM, New York, NY, USA (2017)
5. Brambilla, M., Ceri, S., Celino, I., Cerizza, D., Valle, E.D., Facca, F.M., Turati, A., Tziviskou, C.: Experiences in the design of semantic services using web engineering methods and tools. Journal of Data Semantics (JoDS) 11, 1–31 (2008)
6. Brambilla, M., Ceri, S., Della Valle, E., Volonterio, R., Acero Salazar, F.: Extracting emerging knowledge from social media. In: WWW'17. pp. 795–804 (2017)
7. Brambilla, M., Fraternali, P.: Interaction flow modeling language: Model-driven UI engineering of web and mobile apps with IFML. Morgan Kaufmann, USA (2014)
8. Brambilla, M., Tziviskou, C.: Modeling ontology-driven personalization of web contents. In: 8th International Conference on Web Engineering, ICWE 2008, New York, USA. pp. 247–260 (2008)

[5] http://www.remodd.org/

[6] http://www.mdeforge.org/

9. Brambilla, M., Tziviskou, C.: An online platform for semantic validation of UML models. In: 9th Int.l Conf. on Web Engineering (ICWE) 2009 San Sebastián, Spain. pp. 477–480. Springer (2009)
10. Gasevic, D., Djuric, D., Devedzic, V.: Model Driven Engineering and Ontology Development (2. ed.). Springer (2009), https://doi.org/10.1007/978-3-642-00282-3
11. Gogolla, M., Cabot, J.: Continuing a benchmark for UML and OCL design and analysis tools. In: Workshop on Software Technologies: Applications and Foundations. pp. 289–302 (2016)
12. Haim, S., Walsh, T.: Restart strategy selection using machine learning techniques. In: Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing. pp. 312–325. SAT '09, Springer-Verlag, Berlin, Heidelberg (2009)
13. Hartmann, T., Moawad, A., Fouquet, F., Le Traon, Y.: The next evolution of mde: a seamless integration of machine learning into domain modeling. Software & Systems Modeling (May 2017)
14. Hutchinson, J.E., Whittle, J., Rouncefield, M.: Model-driven engineering practices in industry: Social, organizational and managerial factors that lead to success or failure. Sci. Comput. Program. 89, 144–161 (2014), http://dx.doi.org/10.1016/j.scico.2013.03.017
15. Hutter, F., Babic, D., Hoos, H.H., Hu, A.J.: Boosting verification by automatic tuning of decision procedures. In: Proceedings of the Formal Methods in Computer Aided Design. pp. 27–34. FMCAD '07, IEEE Computer Society, Washington, DC, USA (2007)
16. Hutter, F., Xu, L., Hoos, H.H., Leyton-Brown, K.: Algorithm runtime prediction: Methods & evaluation. Artif. Intell. 206, 79–111 (Jan 2014)
17. Izquierdo, J.L.C., Cabot, J.: Jsondiscoverer: Visualizing the schema lurking behind JSON documents. Knowl.-Based Syst. 103, 52–55 (2016), https://doi.org/10.1016/j.knosys.2016.03.020
18. Kelly, K.: The inevitable: understanding the 12 technological forces that will shape our future. Viking (2016)
19. Kessentini, M., Sahraoui, H.A., Boukadoum, M., Benomar, O.: Search-based model transformation by example. Software and System Modeling 11(2), 209–226 (2012), https://doi.org/10.1007/s10270-010-0175-7
20. Kotthoff, L., Gent, I.P., Miguel, I.: An evaluation of machine learning in algorithm selection for search problems. AI Commun. 25(3), 257–270 (Aug 2012)
21. Kuschke, T., Mäder, P., Rempel, P.: Recommending auto-completions for software modeling activities. In: Proceedings of the 16th International Conference on Model-Driven Engineering Languages and Systems - Volume 8107. pp. 170–186. Springer-Verlag New York, Inc., New York, NY, USA (2013)
22. Lopez-Martin, C., Chavoya, A., Meda-Campaa, M.E.: A machine learning technique for predicting the productivity of practitioners from individually developed software projects. In: SPND'2014. pp. 1–6 (2014)
23. Mazak, A., Wimmer, M.: Towards liquid models: An evolutionary modeling approach. In: Business Informatics (CBI), 2016 IEEE 18th Conference on. vol. 1, pp. 104–112. IEEE (2016)
24. Pati, T., Feiock, D.C., Hill, J.H.: Proactive modeling: Auto-generating models from their semantics and constraints. In: Proceedings of the 2012 Workshop on Domain-Specific Modeling. pp. 7–12. DSM '12, ACM, New York, NY, USA (2012)
25. Perini, A., Susi, A., Avesani, P.: A machine learning approach to software requirements prioritization. IEEE Transactions on Software Engineering 39(4), 445–461 (April 2013)
26. Shepperd, M., Bowes, D., Hall, T.: Researcher bias: The use of machine learning in software defect prediction. IEEE Transactions on Software Engineering 40(6), 603–616 (June 2014)

27. Sottet, J., Ganneau, V., Calvary, G., Coutaz, J., Demeure, A., Favre, J., Demumieux, R.: Model-driven adaptation for plastic user interfaces. In: Human-Computer Interaction - INTERACT 2007, 11th IFIP TC 13 International Conference, 2007, Proceedings, Part I. pp. 397–410 (2007)
28. Steimann, F., Ulke, B.: Generic model assist. In: Proceedings of the 16th International Conference on Model-Driven Engineering Languages and Systems - Volume 8107. pp. 18–34. Springer-Verlag New York, Inc., New York, NY, USA (2013)
29. Stol, K.J., Fitzgerald, B.: Two's company, three's a crowd: A case study of crowdsourcing software development. In: ICSE 2014. pp. 187–198. ACM (2014)
30. Wen, J., Li, S., Lin, Z., Hu, Y., Huang, C.: Systematic literature review of machine learning based software development effort estimation models. Information and Software Technology 54(1), 41 – 59 (2012)