Automatic Support for Multi-Domain Model Management

Weslley Torres, Mark G. J. van den Brand, Alexander Serebrenik
Eindhoven University of Technology, Eindhoven, The Netherlands
{w.silva.torres, m.g.j.v.d.brand, a.serebrenik}@tue.nl

***Abstract*—The process of developing complex systems often involves knowledge of engineers from multiple domains: e.g., to develop a robot one needs to combine expertise about mechanics, electronics, and software. Such domain-specific knowledge is often represented in a form of interdependent models, consequently a change in a model of one domain might impact a model from a different domain. Thus, identifying which models are affected due to a change is an important problem, which is further exacerbated due to heterogeneity of modeling notations used.**

The aim of this PhD research project is to facilitate model management in a multi-domain setting. In the earlier stage of this study, we investigated the available approaches used to manage models from different domains. We concluded that the available approaches are tool-dependent, and do not fully support co-evolution of the models. Additionally, previous research recommends to explicitly indicate the dependency between models in order to support the co-evolution of models from different domains. Since these models are created using different modeling notations we believe that it is not reasonable to develop a tool to parse every notation. Furthermore, it is possible that the source code of the model is missing, but engineers still have an image of the model. Thus, to ensure the maintenance of multi-domain systems we investigated the suitability of optical character recognition (OCR) as a uniform approach. We observed that even though OCR has shortcomings, it produces satisfactory results, and once the identified shortcomings are addressed, OCR can become a crucial technology to support the evolution of multi-domain systems. To this end we envision the development of an infrastructure where we can use OCR to identify relationships between models from different domains, store them in a structured manner making it easier to maintain the consistency of the entire system.

I. Introduction

Models are the primary artifacts in model-driven approaches [1]. Model-driven engineering has been used in fields, such as software engineering [2], robotics [3], and automotive [4]. Managing models that belong to the same domain might not be a complex task due to the features provided by the available development tools. However, managing interrelated models of different domains is challenging [5]. A robot is an example of a such multi-domain system. To develop it one needs to combine models created by experts from mechanics, electronics and software. These models might be created using domain specific tools of each domain, and a change in one model of one domain might impact a model from a different domain causing inconsistency in the entire system.

This PhD research project therefore proposes to facilitate the evolution of the models in this multi-domain setting. The initial phase of this study consisted of a systematic literature review [6] in order to identify the open issues, and strategies used to manage models from different domains. As result, we identified that making explicit the relationship between models from different domains can support the maintenance of the models, making it easy to identify affected models due to a change. The following step was to investigate ways of extracting information from different engineering models that were created using different modeling notations. Due to the amount and diversity of existing modeling notations, we considered that it is not reasonable to develop a tool capable of parsing all types of models. Even if we develop such a tool, it would have to be updated every time that a new notation emerges. Moreover, these models might only be available as pictures and not as a structured format, e.g., XML. Thus, we need a uniform approach that would be independent from the peculiarities of the notation. This uniform approach can only be based on elements typically present in various modeling notations, i.e., text, boxes, and lines.

We investigated the suitability of optical character recognition (OCR) for extracting textual elements from models from different domains [7]. We compared the accuracy of two off-the-shelf OCR services (Google Cloud Vision and Microsoft Cognitive Services) in extracting textual elements from a collection of 43 models from different domains. Additionally, we identified the common errors made by Google Cloud Vision and we observed that even though OCR has some limitations, such as not being able to detect mathematical equations, Greek letters and misinterpreting one textual element as two separate elements due to the text being positioned on multiple lines. It produces satisfactory results being able to detect 70% of the textual elements. We believe that once the main shortcomings are addressed, OCR can become a crucial technology to support multi-domain model management. In the ongoing work we are improving the precision and recall addressing the main OCR limitations. Preliminary results indicate that we corrected up to 50% of the textual elements that were misinterpreted due to positioning on multiple lines.

The next step is to automatically detect relationships between models from different domains. To this purpose, we are going to use OCR, name matching, and image processing to detect shapes, such as boxes, arrows and lines.

It is common in modeling languages to use boxes to group elements, and lines connecting these boxes to indicate the presence of relationships. The UML class diagram in software engineering domain is an example of such representation. Therefore, we are going to use image process techniques

to detect shapes presented in a variety of models and infer possible relationships.

Our hypothesis is that elements might share the same name even though they are in models from different domains, therefore, identifying relations between them is useful to support the modeling process. We are going to use name matching on the textual elements extracted using OCR to evaluate this hypothesis. We found promising results in our preliminary experiments to confirm this hypothesis. Our second hypothesis is that detecting boxes and lines/arrows presented on the models can be used to provide additional relationships to the ones found using the name matching.

To conclude the project, we envision the development of an infrastructure that combines all the previous elements into one single tool that can also store the relationships in a structured manner making it easier to maintain the consistency of an entire system. The development of this infrastructure is in an advanced stage and we plan to evaluate it by means of an observational study with a multidisciplinary team that builds autonomous robots designed to play football.

II. Related Work

A. Explicit Dependency Modeling

Sirin et al. [8] use DSM (Design Structure Matrix) and DMM (Design Multiple Matrix) to identify and map the dependencies between models. The drawback about DSM and DMM is that, it is not possible to describe the semantics of the relationships. Qamar et al. [9] investigated the dependencies between models and how to model these dependencies. Törngren et al [10] extended that work and proposed an approach to integrate different viewpoints using Dependency Modeling Language (DML) and Dependency Modeler [9].

Dávid et al. [11], [12], [13] propose a modelling language that is capable of modelling the process and the properties of the system. This language allows engineers to describe relationships between properties and the level of precision. Feldmann et al. [14], [15] propose specify and manage inter-model inconsistency. In order to maintain consistency during model evolution, they create links between the dependent entities and define rules to enable the consistency checking.

Daniel et al. [16] propose a model persistence framework (NeoEMF), as a set of Eclipse plugins, capable of storing models focusing on model transformation. Didonet Del Fabro et. al. [17] propose weaving model as a mapping between elements of two (meta)models. In order to perform this mapping, they created an Eclipse plug-in called Atlas Model Weaver (AMW) making their approach tool dependent.

B. Optical Character Recognition (OCR)

To the best of our knowledge, there are no studies on the use of off-the-shelf OCR services in the extraction of text on models from different domains.

Recently, researchers evaluate the accuracy of six OCR engines (Google Drive OCR, ABBYY FineReader, GOCR, Tesseract, OCRAD, and Cuneiform) in extracting code (Java, Python, C#) from screencasts and code images [18]. They conclude that although Google Drive OCR and ABBYY FineReader are slower than the others, they are more accurate.

Dataturks [19] presents a comparison between three OCR services (Amazon AWS, Google Cloud Vision, and Microsoft Cognitive). They measured the precision and recall in recognizing text from a random sample of 500 images of business names or movie names. They conclude that Google Cloud Vision outperformed the others.

Reis [20] compare Google Cloud Vision and Microsoft Cognitive Services in recognizing text from the photos of the pages of the Bible. Mello and Dueire Lins [21], and Vijayarani and Sakila [22] published additional comparison studies.

OCR has been used in extract text on domain specific modeling languages, for instance Img2UML [23], [24]. This tool extracts UML Class Diagrams from images. Additionally, there are studies that have used OCR as part of a tool classifying images as UML diagrams: targeting class diagrams [25], [26], sequence diagrams [27] and component diagrams [26].

III. Past Research

A. Systematic Literature Review

We conducted a systematic literature review to investigate the industrial practices and academic approaches to ensuring consistency in cross-domain model management [6] (an extended version of this paper is currently under review).

a) Methodology: In this systematic literature review, we followed strict guidelines proposed by Kitchenham and Charters [28], we used Google Scholar as the search engine [29], [30], [31], [32]. We create search strings to query Google Scholar, based on PICO [33]. Due to the similarity of the queries, some papers were retrieved multiple times. We automatically excluded these duplicates prior to the manual inspection. In summary, we obtained 3222 hits, with 515 of them unique. We obtained 168 papers when the selection criteria was applied, and we conclude the process with 88 papers. To avoid bias in the selection of the papers, we computed Cohen's κ [34] to measure the agreement between the raters and discussed the disagreements.

b) We have obtained the following results:

RQ) How do model life cycle management tools support consistency checking? We observed that approximately 31% of the tools we found can check the consistency on models from different domains, approximately 24% on the models of the same domain, and approximately 45% do not provide any consistency checking.

RQ) Which strategies have been used to keep the consistency between models of different domains? We organized the identified strategies into categories that include the use of a extensible catalogue of inconsistency patterns, constraints management, modeling dependencies explicitly, ontologies, and standard data exchange. However, we observed that these strategies are tool-dependent, and do not fully support co-evolution of the models. For example, they often fail to notify affected models due to a change, to maintain the history of changes and to distinguish between different types of relationships.

B. Suitability of OCR to Support Model Management

To ensure consistency Qamar et al. [9] recommend explicit modeling of the relationships between models. A number of technologies can be used to model these relationships explicitly [10], [9], [14], [15]. However, little is known about approaches to automatically identify the relationships between models from different domains. The main challenge is the heterogeneity of modeling notations: we believe that it would not be feasible to develop a tool to parse all the existing notations. Moreover, even if such a tool was developed, it would have to be updated every time a modeling notation evolves or a new notation emerges. Thus, we need a uniform approach based on elements present in all those models, i.e., text, boxes, and lines. Moreover, some tools might not be able to export the models into a intermediate structured format, such as XMI. Another possibility is the model only being available as an image, either because the source code of the model is missing, or because the engineer only have the model as a physical paper document.

Thus, we investigated the suitability of OCR to be this uniform approach independent from the peculiarities of the notation [7]. Following are the research questions used to guide this study:

RQ) How accurate are off-the-shelf OCR services for extracting text from graphical models? To answer this research question, we applied Google Cloud Vision and Microsoft Cognitive Services to a collection of 43 models from different domains. These models were selected from two UML open repositories [35], [36], three control system engineering papers [37], [38], [39], and the example catalog of MatLab Simulink[1]. We observed that Google Cloud Vision outperformed Microsoft Cognitive Services being able of correctly recognize 854 out of 1,232 textual elements, while Microsoft Cognitive Services correctly recognized 388 elements. In terms of precision and recall, Google Cloud Vision outperformed Microsoft Cognitive Services in the majority of the models.

RQ) What are the common errors made by OCR services on models from different domains? We focused on the common errors from Google Cloud Vision due to performance presented in the previous RQ. We identified 17 common errors and we grouped them into four categories: non-alphanumeric characters, mathematical formulas, spacing, and character confusion. The most common errors are related to spacing (text written on multiple lines, and an empty space between letters), and character confusion (wrong/missing characters). It is also important to stress that Google Cloud Vision failed in detecting textual elements positioned on multiple lines, Greek letters, subscripts, and equations. Due to space limitations text can be positioned on multiple lines, making OCR to misinterpret as one textual element but as two separate element. We named this error as "multi-line text". The fixing of this error was the starting point for the ongoing research detailed in the next section.

[1]https://www.mathworks.com/products/simulink.html

IV. Ongoing Research

This section summarizes our ongoing work, where we aim to (i) improve the precision and recall of the OCR focusing on the main OCR limitations, (ii) detect shapes such as boxes, lines and arrows to be used to group the textual elements and support the identification of relationships between models from different domains, and (iii) develop a tool in which we can identify and store relationships between models from different domains.

Precision and Recall As described in the previous section, we observed that Google Cloud Vision failed to detect textual elements that are positioned in multiple lines. To fix this problem we use a set of heuristics that take into consideration such parameters as the alignment, the distance between the words, and the size of the words and letters. Preliminary results indicate that applying these heuristics can fix up to 50% of the textual elements that were misinterpreted due to the text being positioned on multiple lines.

Shape Detection It is common for engineers to use shapes, such as boxes to group elements, and lines connecting these boxes to indicate the presence of relationships between them. The UML class diagram in software engineering domain is an example of such representation. Hence, we are using a set of image processing techniques provided by OpenCV[2] to automatically detect shapes presented in the models. We believe that the shape detection can support in the identification of relationships between models from different domains.

Our first hypothesis is that elements might share the same name even though they are in models from different domains, consequently establishing some relation between them. Therefore, we are going to use name matching on the textual elements extracted using OCR to evaluate this hypothesis. We found promising results in our preliminary experiments. Our second hypothesis is that shapes, such as boxes and arrows presented on the models can be used to provide additional relationships to the ones found using the name matching. For this research we aim at identify the relationships between models from different domains. The identification of the type of these relationships is out of the scope.

Tooling To conclude the project, we envision the development of an infrastructure that combines OCR, name matching, and shape detection previously described into one single tool. With this tool, it will be possible to identify and store the relationships between models from different domains in a structured manner, making it easier to maintain the consistency of the entire system. The development of this infrastructure is in advanced stage and we plan to evaluate it in an observational study with a multidisciplinary team that builds autonomous robots designed to play football.

V. Conclusion

The main goal of this research is to support the (co-)evolution of models from different domains. So far, we conducted a systematic literature review to identify open issues

[2]https://opencv.org

related to consistency management in multi-domain modeling. The outcome of this literature review served as a basis to the follow-up study where we investigated the suitability of OCR for extracting information from models. Currently, we are improving the accuracy of OCR, and we are using image processing techniques to detect shapes, such as boxes and lines presented on models. Thus, we believe that using the text extracted by OCR, shapes, and name matching, we can infer possible relationships between the models.

REFERENCES

[1] M. Herrmannsdörfer and G. Wachsmuth, "Coupled evolution of software metamodels and models," in *Evolving Software Systems*. Springer, 2014, pp. 33–63.

[2] C. Atkinson, "Orthographic software modelling: a novel approach to view-based software engineering," in *ECMFA*, ser. LNCS. Springer, 2010, vol. 6138, pp. 1–1.

[3] Y. Sun, J. Gray, K. Bulheller, and N. von Baillou, "A model-driven approach to support engineering changes in industrial robotics software," in *MODELS*, ser. LNCS. Springer, 2012, vol. 7590, pp. 368–382.

[4] S. Mustafiz, J. Denil, L. Lúcio, and H. Vangheluwe, "The ftg+ pm framework for multi-paradigm modelling: An automotive case study," in *Proceedings of the 6th International Workshop on Multi-Paradigm Modeling*, 2012, pp. 13–18.

[5] R. Hebig, H. Giese, F. Stallmann, and A. Seibel, "On the complex nature of mde evolution," in *MODELS*, ser. LNCS. Springer, 2013, vol. 8107, pp. 436–453.

[6] W. Torres, M. G. J. van den Brand, and A. Serebrenik, "Model management tools for models of different domains: A systematic literature review," in *SysCon*, 2019.

[7] ——, "Suitability of optical character recognition (ocr) for multi-domain model management," in *International Conference on Systems Modelling and Management*. Springer, 2020.

[8] G. Sirin, B. Yannou, E. Coatanéa, and E. Landel, "Analyze of the simulation system in an automotive development project," 2012.

[9] A. Qamar, C. J. Paredis, J. Wikander, and C. During, "Dependency modeling and model management in mechatronic design," *Journal of Computing and Information Science in Engineering*, vol. 12, no. 4, p. 041009, 2012.

[10] M. Törngren, A. Qamar, M. Biehl, F. Loiret, and J. El-Khoury, "Integrating viewpoints in the development of mechatronic products," *Mechatronics*, vol. 24, no. 7, pp. 745–762, 2014.

[11] I. Dávid, J. Denil, K. Gadeyne, and H. Vangheluwe, "Engineering process transformation to manage (in) consistency," in *Proceedings of COMMitMDE 2016*, 2016, pp. 7–16.

[12] I. Dávid, B. Meyers, K. Vanherpen, Y. Van Tendeloo, K. Berx, and H. Vangheluwe, "Modeling and enactment support for managing inconsistencies in heterogeneous systems engineering processes," in *Proceedings of MODELS 2017*, 2017, pp. 145–154.

[13] I. Dávid, J. Denil, and H. Vangheluwe, "Process-oriented inconsistency management in collaborative systems modeling," in *16th Industrial Simulation Conference (2018)*, 2018, pp. 54–61.

[14] S. Feldmann, M. Wimmer, K. Kernschmidt, and B. Vogel-Heuser, "A comprehensive approach for managing inter-model inconsistencies in automated production systems engineering," in *CASE*. IEEE, 2016, pp. 1120–1127.

[15] S. Feldmann, K. Kernschmidt, M. Wimmer, and B. Vogel-Heuser, "Managing inter-model inconsistencies in model-based systems engineering: Application in automated production systems engineering," *Journal of Systems and Software (2019)*, vol. 153, pp. 105–134, 2019.

[16] G. Daniel, G. Sunyé, A. Benelallam, M. Tisi, Y. Vernageau, A. Gómez, and J. Cabot, "Neoemf: a multi-database model persistence framework for very large models," *Science of Computer Programming*, vol. 149, pp. 9–14, 2017.

[17] M. D. Del Fabro, J. Bézivin, F. Jouault, E. Breton, and G. Gueltas, "Amw: a generic model weaver," in *1 ere Journées sur l'Ingénierie Dirigée par les Modèles (2005)*.

[18] A. Khormi, M. Alahmadi, and S. Haiduc, "A study on the accuracy of ocr engines for source code transcription from programming screencasts," 06 2020.

[19] "Image text recognition apis showdown," https://dataturks.com/blog/compare-image-text-recognition-apis.php, accessed: 2020-01-08.

[20] A. Reis, D. Paulino, V. Filipe, and J. Barroso, "Using online artificial vision services to assist the blind - an assessment of microsoft cognitive services and google cloud vision," in *Trends and Advances in Information Systems and Technologies*. Springer, 2018, pp. 174–184.

[21] C. A. B. Melo and R. Dueire Lins, "A comparative study on ocr tools," in *Vision Interface*, 1999.

[22] S. Vijayarani and A. Sakila, "Performance comparison of OCR tools," *International Journal of UbiComp*, vol. 6, no. 3, pp. 19–30, 2015.

[23] B. Karasneh and M. R. Chaudron, "Extracting uml models from images," in *2013 5th International Conference on Computer Science and Information Technology*. IEEE, 2013, pp. 169–178.

[24] ——, "Img2uml: A system for extracting uml models from images," in *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*. IEEE, 2013, pp. 134–137.

[25] T. Ho-Quang, M. R. Chaudron, I. Samúelsson, J. Hjaltason, B. Karasneh, and H. Osman, "Automatic classification of uml class diagrams from images," in *2014 21st Asia-Pacific Software Engineering Conference*, vol. 1. IEEE, 2014, pp. 399–406.

[26] V. Moreno, G. Génova, M. Alejandres, and A. Fraga, "Automatic classification of web images as uml diagrams," in *Proceedings of the 4th Spanish Conference on Information Retrieval*, 2016, pp. 1–8.

[27] S. Rashid, "Automatic classification of uml sequence diagrams from images," 2019.

[28] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering. engineering 2 (2007), 1051," *arXiv preprint arXiv:1304.1186*, 2007.

[29] D. Landman, A. Serebrenik, and J. J. Vinju, "Challenges for static analysis of Java reflection-literature review and empirical study," in *International Conference on Software Engineering*. IEEE, 2017, pp. 507–518.

[30] F. A. Moghaddam, P. Lago, and P. Grosso, "Energy-efficient networking solutions in cloud-based environments: A systematic literature review," *ACM Comput. Surv.*, vol. 47, no. 4, pp. 64:1–64:32, May 2015.

[31] S. Jalali and C. Wohlin, "Systematic literature studies: Database searches vs. backward snowballing," in *International Symposium on Empirical Software Engineering and Measurement*, Sept 2012, pp. 29–38.

[32] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proceedings of the 18th international conference on evaluation and assessment in software engineering*. ACM, 2014, p. 38.

[33] B. A. Kitchenham, E. Mendes, and G. H. Travassos, "Cross versus within-company cost estimation studies: A systematic review," *IEEE Transactions on Software Engineering*, no. 5, pp. 316–329, 2007.

[34] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, 1960.

[35] "The uml repository," http://models-db.com, accessed: 2020-01-23.

[36] "Git uml repository," https://www.gituml.com, accessed: 2020-01-23.

[37] B. Ai, L. Sentis, N. Paine, S. Han, A. Mok, and C.-L. Fok, "Stability and performance analysis of time-delayed actuator control systems," *Journal of Dynamic Systems, Measurement, and Control*, vol. 138, no. 5, 2016.

[38] S. Tovar-Arriaga, J. E. Vargas, J. M. Ramos, M. A. Aceves, E. Gorrostieta, and W. A. Kalender, "A fully sensorized cooperative robotic system for surgical interventions," *Sensors*, vol. 12, no. 7, pp. 9423–9447, 2012.

[39] V. K. Kaliappan, H. Yong, M. Dugki, E. Choi, and A. Budiyono, "Reconfigurable intelligent control architecture of a small-scale unmanned helicopter," *Journal of Aerospace Engineering*, vol. 27, no. 4, p. 04014001, 2014.