From Conceptual Models to Knowledge Graphs: A Generic Model Transformation Platform

Muhamed Smajevic and Dominik Bork

To appear in:

2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C) – Tools & Demonstrations Track, 2021

Final version available via DOI (to be added once published):

From Conceptual Models to Knowledge Graphs: A Generic Model Transformation Platform

Muhamed Smajevic
TU Wien, Business Informatics Group
Favoritenstrasse 11, 1040 Vienna, Austria
Email: e11742556@student.tuwien.ac.at

Dominik Bork
TU Wien, Business Informatics Group
Favoritenstrasse 11, 1040 Vienna, Austria
Email: dominik.bork@tuwien.ac.at

***Abstract*—Semantic processing of conceptual models is a focus of research since several years, bridging the disciplines of knowledge-based systems, conceptual modeling, and model-driven software engineering. With the uptake of Knowledge Graphs, the research in this area gained further momentum. In this paper, we introduce a generic and extensible platform that enables the automated transformation of conceptual models into Knowledge Graphs. The platform can transform any model created by a state-of-the-art metamodeling platform (EMF and ADOxx) into standardized Knowledge Graph representations like GraphML, RDF, and OWL. In the paper at hand, we introduce our platform and evaluate it with a corpus of 5.000 UML models that we transform into Knowledge Graphs and subsequently exemplify the rich functionalities enabled by the graph structure by an automated detection of UML model smells.**

***Index Terms*—Knowledge Graph, Modeling tool, Code smells, Model refactoring, Model transformation**

I. Introduction

Semantic processing of conceptual models is a focus of research since several years, bridging the disciplines of knowledge-based systems, conceptual modeling, and model-driven software engineering (MDSE). While manual analysis might be feasible for small models, automation is required for medium to large-size models which we often find in practice. The aim of such an automated analysis is to *"extract valuable information"* from conceptual models *"to support the analytic process."* [1, p. 1] Recently, first ideas have been proposed to use Knowledge Graphs for reasoning on conceptual models [2] and to transform conceptual models into Knowledge Graphs – as e.g., reported for Genomic Datasets by Bernasconi et al. [3]. A Knowledge Graph is *"a large network of entities, and instances for those entities, describing real world objects and their interrelations, with specific reference to a domain or to an organization."* [4, p. 27] A Knowledge Graph encapsulates a knowledge base, which is represented by a graph structure, and a reasoning component that enables to reason about the knowledge represented by the graph.

This paper builds upon and extends our previous works that focused on the graph-based analysis of enterprise architecture models [5]. In the paper at hand, we introduce a generic model transformation platform – Conceptual Model to Knowledge Graph (CM2KG) – that is capable of transforming any conceptual model realized with state-of-the-art metamodeling platforms like EMF, ADOxx, or Papyrus (for UML, SysML and UML profile models). The CM2KG platform thus excels existing works on graph-based analysis of conceptual models by being *generic* (i.e., modeling language and modeling tool independent), *extensible* (i.e., new language- and tool-specific transformations can be easily realized), and *scalable* (i.e., even very large Knowledge Graphs can be efficiently processed [4]).

As a running example, we show the transformation of conceptual models of the Unified Modeling Language (UML) into Knowledge Graphs represented in GraphML (i.e., the knowledge base). To exemplify the value of the transformation – or more particularly the value of the Knowledge Graph in MDSE – we show how the graph structure enables efficient reasoning by means of graph queries. This reasoning aims to automatically detect software design smells. Using a corpus of 5.000 publicly available UML models [6], we evaluate the validity of the transformation and the efficiency of the reasoning applied onto the transformed Knowledge Graphs.

II. The CM2KG Cloud Platform

In this section, we report on the realization of the CM2KG Cloud platform. Fig. 1 shows the framework that forms the foundation of the platform. CM2KG is cloud-based, thus offering the transformation, visualization, and analysis functionality through the web browser. The implementation and the example models can be found in the accompanying Github repository [7].

A. CM2KG Platform Architecture

The architecture of the CM2KG platform consists of three components: *Model Import*, *CM2KG Cloud*, and *Third Party Tools*. In the model import component, users upload a model (in Extensible Markup Language (XML) format) which they have created (and exported) with any EMF- or ADOxx-based modeling tool including Archi and Papyrus. The XML file serves as the input for the *CM2KG Cloud* component. CM2KG Cloud consists of two virtual machines running the web server and the database, respectively. The web server serves the web application and the model transformation service, while the Neo4j database stores the transformed Knowledge Graph and offers additional functionality. The front-end is realized with the neovis.js [8] library, which provides a rich set of graph visualizations (see Fig. 2b).

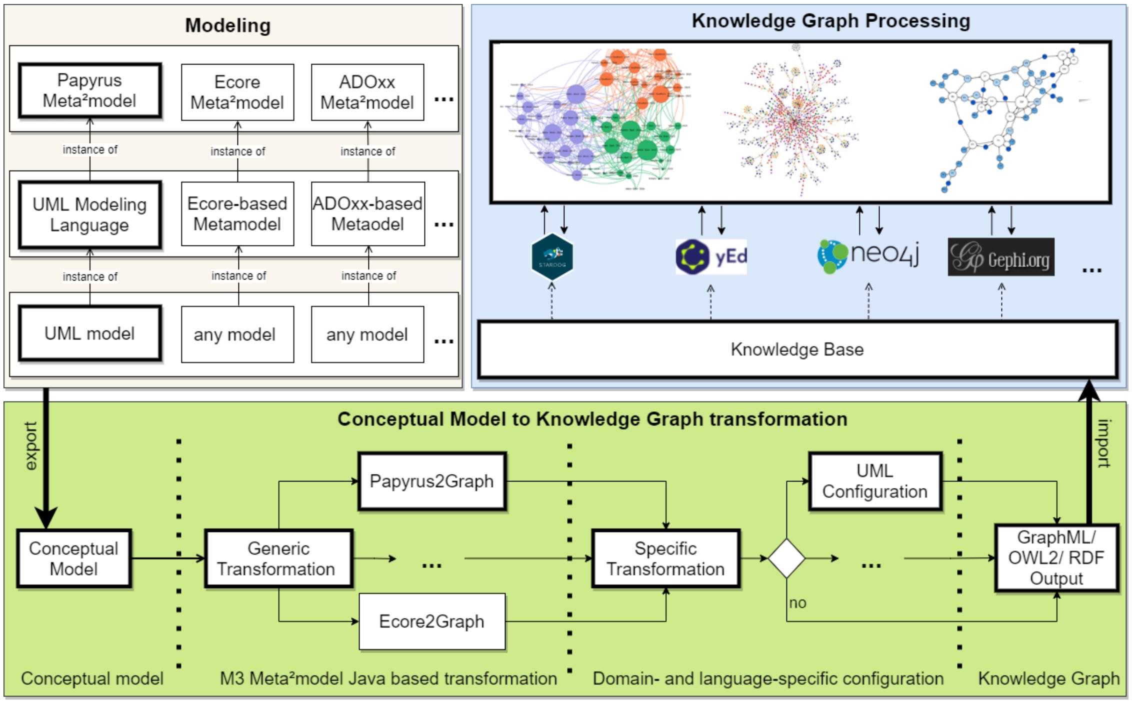


Fig. 1. Conceptual Model to Knowledge Graph (CM2KG) Cloud platform.

B. CM2KG Core Functionalities

In the following, we briefly introduce the core functionality of the CM2KG Cloud platform. A more detailed introduction to the technical realization of the transformation is not possible given the limited space. A detailed discussion of the transformation from Ecore models to GraphML Knowledge Graphs can be found in [5].

1) Model Transformation: The CM2KG Cloud platform offers the functionality to upload a conceptual model in XML format and to transform it into a Knowledge Graph which can be inspected in the browser or directly downloaded. Currently, the platform supports the transformation of any conceptual model created with the EMF or ADOxx metamodeling platforms as well as the Ecore-based modeling platforms Papyrus (for UML, SysML, and UML profiles) and Archi (for ArchiMate).

2) Knowledge Base Initialization: The CM2KG Cloud platform not only transforms a given conceptual model into a Knowledge Graph, it moreover directly initializes a knowledge base with the resulting graph in Neo4j. This database instance is used for the visual inspection of the graph (discussed in the following). The database reference also allows third-party graph analysis tools to directly connect to the instance.

If further analysis is required, CM2KG Cloud supports the export of the Knowledge Graph in GraphML, OWL2, and RDF format. These standardized formats can then be used as an input for any dedicated graph analysis third party tools [9] like Gephi, yEd, or Stardog.

3) Knowledge Graph Visualisation & Analysis: The CM2KG Cloud platform visualizes the Knowledge Graph in the central pane (area 3 in Fig. 2b). The platform surrounds this pane with powerful visualization and analysis functionality, e.g., to customize the rendering of the graph, i.e., with respect to size, color, and labels of the graph elements (area 4 in Fig. 2b). CM2KG supports three different means of analyzing the Knowledge Graph directly in the cloud:

Predefined analysis queries The user may execute standardized structural graph analysis queries such as *centrality* and *community detection* queries (area 1 in Fig. 2b). These queries are presented to the user to enable direct adaptation (area 2 in Fig. 2b).

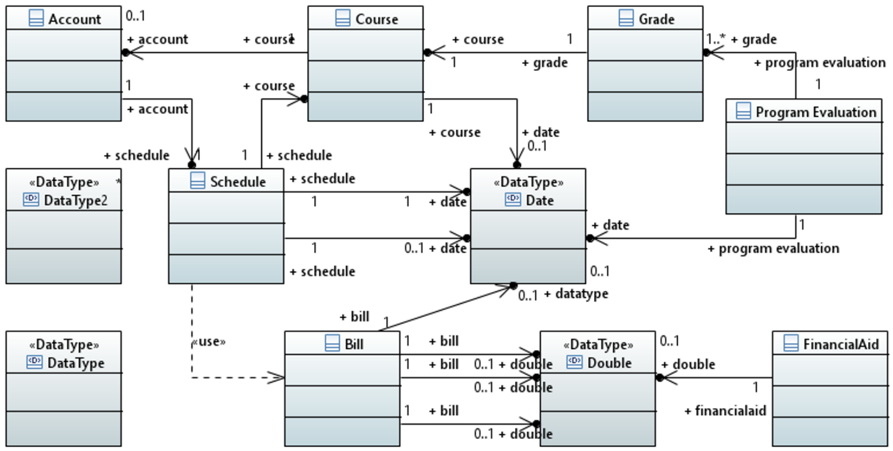
Code smell analysis queries The user may execute some of the code smell queries introduced in this paper.

User-defined queries The user may also directly run any complex Cypher query by using the query text field on the lower right area of the CM2KG Cloud platform (see Fig. 2b, area 5).

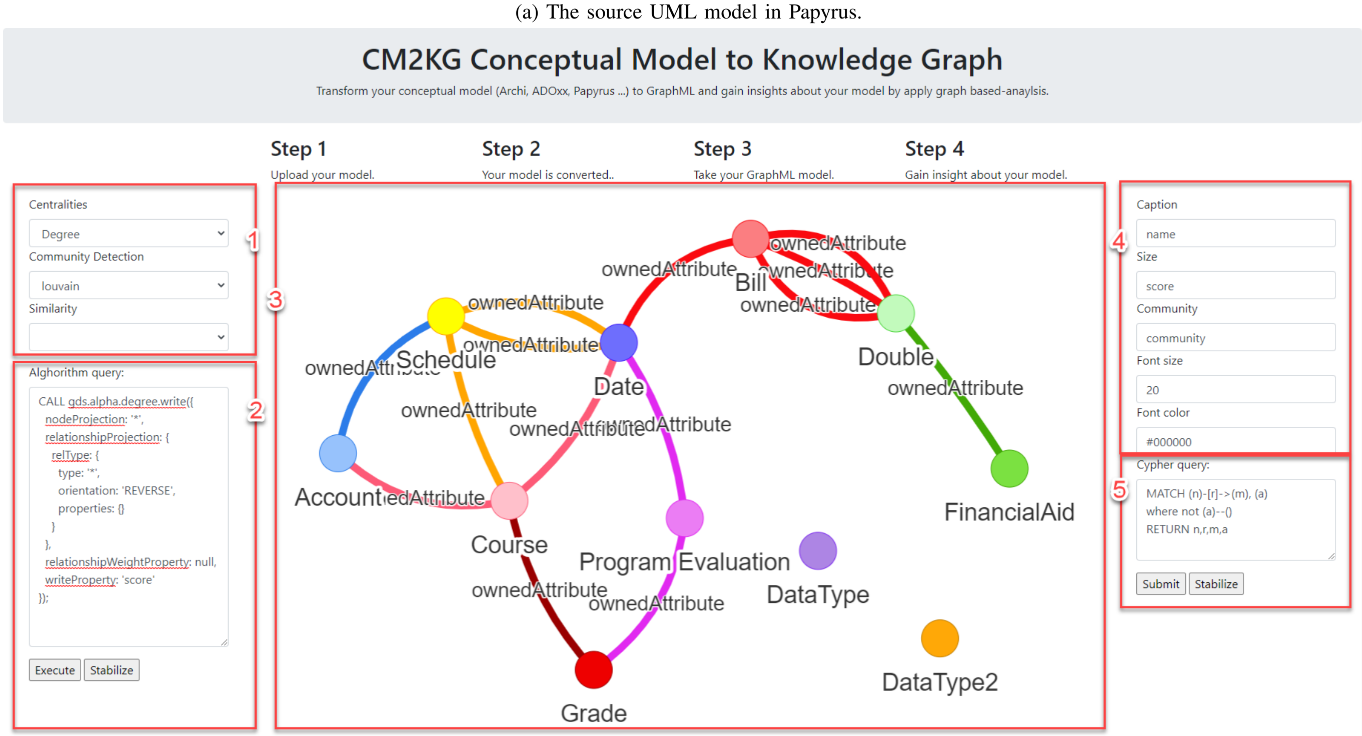
The execution of a query always triggers an update of the visualized graph, i.e., CM2KG directly updates the central pane (area 3 in Fig. 2b) to render the result of the query directly in the browser. Likewise, each change in the visualization configuration will trigger an UI update.

III. Case Study: Knowledge Graph based Code Smell Detection

The detection of code smells is not a new research direction in itself. Early works go back to Fowler [10] who published a well respected book on refactorings in 1999. Afterwards, others further developed code smells and mapped them to UML Class Diagrams [11]–[14]. It needs to be stated, that this



(a) The source UML model in Papyrus.



(b) The transformed Knowledge Graph in the CM2KG Cloud platform.

Fig. 2. An example transformation from a UML model (a) to a Knowledge Graph (b) in the CM2KG Cloud platform.

paper does not intend to make a contribution on code smells or code refactorings. Instead, we use the code smell detection as a relevant and ongoing research field in MDSE. We aim to show the feasibility and strengths of transforming conceptual models into Knowledge Graphs, thereby preparing them for semantics analysis and reasoning. The realized Knowledge Graph queries though make a minor technical contribution by themselves.

Table I describes five UML smells summarized by Haendler [13] based on previous works by Fowler [10] and Suryanarayana et al. [12]. This set of metrics serves as an initial feasibility assessment of our platform. We show, how they can be i) realized on a Knowledge Graph, and ii) executed on a large corpus of UML models [6]. The selection of smells aims to cover the diversity of design smells that can be detected solely based on the UML Class Diagram[1]: *Hierarchy Smells* (Deep Hierarchy and Multipath Hierarchy), *Abstraction Smells* (Unutilized Abstraction), and *Modularization Smells* (Cyclic Dependency and Message Chain).

IV. Evaluation

For evaluating the CM2KG platform, we first transformed 5.025 UML Class Diagram models which were created with

[1]Note that other smells like *Deficient Encapsulation* additionally require e.g., to consider the corresponding UML Sequence Diagram.

TABLE I
KNOWLEDGE GRAPH QUERIES DETECTING EA SMELLS

UML Class Diagram Smell – based on [10], [12], [13]	**Knowledge Graph Query** – realized as a Neo4j Cypher query
Cyclic Dependency Two or more units (e.g., classes, methods) are mutually dependent.	`MATCH (a)-[*]->(a)` `return a, path`
Message Chain A client unit (e.g., method) calls another unit, which then in turn calls another unit, and so on (navigation through class structure).	`MATCH (a)-[r1]->(b)-[r2]->(c)-[r3]->(d)-[r4]->(e)` `return a,r1,b,r2,c,r3,d,r4,e`

Unutilized Abstraction
Not or barely used units (e.g., class or method).
	`MATCH (n)`
	`WHERE n.isAbstract="true" and not (n)--()`
	`return n`

Deep Hierarchy
An unnecessarily deep hierarchy.
	`MATCH (a)-[r1]->(b)-[r2]->(c)-[r3]->(d)`
	`where r1.Label ='generalization' and r2.Label ='generalization' and r3.Label ='generalization'`
	`return a,b,c,d,r1,r2,r3`

Multipath Hierarchy
A subtype inherits both directly and indirectly from a supertype.
	`MATCH (c)<-[r3]-(a)-[r1]->(b)-[r2]->(c)`
	`where r1.Label ='generalization' and r2.Label ='generalization' and r3.Label ='generalization'`
	`return a,b,c,r1,r2,r3`

the Papyrus [15] modeling tool and made available through the MAR search engine [6] before we analyzed them to automatically detect UML model smells. We applied the realized UML Smells Knowledge Graph queries defined in Table I. The evaluation aimed to respond to the following research questions:

RQ.1 – Feasibility Is our approach feasible to automatically detect code smells in UML models?

RQ.2 – Quality How many code smells can we identify in an openly available corpora of UML models?

RQ.3 – Performance How fast is our Knowledge Graph based approach in detecting code smells in UML models?

In order to run the evaluation an experiment was set up in custom manner to process folders containing many models while using the CM2KG functionality. In the future, we plan to have such a possibility directly integrated in the UI of the platform. From an initial set of 5.025 UML Class Diagrams, the platform successfully transformed and analyzed 4.262 models (84.8%), the remaining 763 models were excluded because they contain special characters, hampering their upload into the graph database engine. This can be easily fixed in a preprocessing step but we can still positively respond to the feasibility of our approach (RQ.1).

The results of analyzing the remaining 4.262 models (RQ.2) in order to detect code smells can be seen in Fig. 3. The analysis showed that all smells mentioned in Table I were found in the model repository. In total, 548 out of 4.262 models had at least one code smell with *Message Chain* being the most frequent one, followed by *Deep Hierarchy*. 178 models had two or more code smells, while the most frequent one (102 co-occurrences) we found was the combination of *Message Chain* and *Deep Hierarchy*, followed by *Cyclic Dependency* and *Message Chain* (62 co-occurrences).

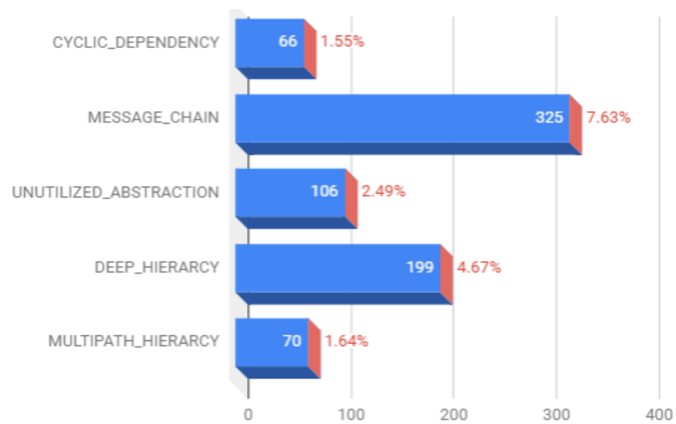


Fig. 3. Code smells detected in 4.262 UML Class Diagram models.

Overall, the transformation and analysis (i.e., the execution of the queries to detect smells) lasted around 30 minutes. The average transformation time was 16ms while the longest transformation lasted 14s. On the other side, the queries had an average execution time ranging from 0.007s to 0.43s. It is important to note, that some transformed models were quite large with more than 100 MB in size, resulting in a huge impact on the time. Considering the Knowledge Graph size, the average number of nodes was 12 (±1.853), the average number of edges was 7 (±111). The largest graph comprised 2.044 nodes and 129 edges. In the future, we plan to investigate the impact of model size on the transformation and analysis performance. For now, we can state that the results are promising with respect to the performance (RQ.3).

V. Related Work

Graph-based approaches have a long tradition in MDSE, especially in the field of model transformation using graph grammars [16]. Also, research on the transformation of UML models into graphs has been conducted for a while, see e.g., [17]. Graph-based analysis is also widely adopted in enterprise modeling and enterprise architecture, see e.g. [18], [19]. The use of Knowledge Graphs is a very recent development with first domain-specific applications proposed in [2], [3], [5], [20].

Several tools exist that are capable to detect code smells in Java source code or UML Class Diagrams, e.g., EMF Refactor, DECOR [21] and JDeodorant [22] – see [13] for a recent survey. Most of them use static analysis techniques that are only applicable to Java-based programs (or only few languages) [13]. What is missing, and what we aim to contribute with the CM2KG Cloud platform, is an environment that is language agnostic, easy to adapt to new languages in case of language-specific smells, and scalable to analyze large models. In contrast to our platform, however, some tools not only detect smells but also provide means to refactor the code. Conversely, CM2KG provides all the functionality in the web, thereby mitigating installation and compatibility issues.

The aim of this paper was not to contribute to code refactoring research, but, considering the first insights we gained through the case study, we consider collaborating with code refactoring experts to continue working on this stream.

VI. Conclusion

In this paper, we presented the CM2KG Cloud platform – a generic, extensible, and scalable platform for the transformation of conceptual models into Knowledge Graphs using model transformation. Our approach is generic in the sense of being realized on the meta-metamodel level (i.e., not dependent on one modeling tool) and extensible to ease its adoption for other conceptual modeling languages while the Knowledge Graph promotes scalability. We reported on the conceptual architecture and the implementation of our platform.

Besides introducing the core aspects of the CM2KG Cloud platform, we exemplified its value by transforming and analyzing a corpus of 5.000 UML Class Diagram models. The use case proved feasibility, efficiency, and scalability of the approach and yielded insights into the code smells prevalent in the corpus. Our major research is not focusing on code smells but instead on providing the generic CM2KG Cloud platform that the model-driven software engineering community can use and extend to plug-in their specific analysis and reasoning algorithms. CM2KG is therefore freely available [7]. In our future research, we will develop further use cases for the Knowledge Graph based reasoning on conceptual models.

References

[1] A. Santana, K. Fischbach, and H. Moura, "Enterprise architecture analysis and network thinking: A literature review," in *49th Hawaii Int. Conference on System Sciences*. IEEE, 2016, pp. 4566–4575.

[2] D. Medvedev, U. Shani, and D. Dori, "Gaining insights into conceptual models: A graph-theoretic querying approach," *Applied Sciences*, vol. 11, no. 2, p. 765, 2021.

[3] A. Bernasconi, A. Canakoglu, and S. Ceri, "From a conceptual model to a knowledge graph for genomic datasets," in *Conceptual Modeling - 38th International Conference, Proceedings*, Laender et al., Ed. Springer, 2019, pp. 352–360.

[4] L. Bellomarini, D. Fakhoury, G. Gottlob, and E. Sallinger, "Knowledge graphs and enterprise ai: the promise of an enabling technology," in *35th International Conference on Data Engineering*. IEEE, 2019, pp. 26–37.

[5] M. Smajevic and D. Bork, "Towards graph-based analysis of enterprise architecture models," in *Proceedings of the 40th International Conference on Conceptual Modeling*, 2021, p. in press.

[6] J. A. H. López and J. S. Cuadrado, "MAR: a structure-based search engine for models," in *ACM/IEEE 23rd International Conference on Model Driven Engineering Languages and Systems*, E. Syriani, H. A. Sahraoui, J. de Lara, and S. Abrahão, Eds. ACM, 2020, pp. 57–67.

[7] D. Bork and M. Smajevic, "Source code repository of the CM2KG cloud platform," https://github.com/borkdominik/CM2KG, 2021.

[8] Neo4j Contrib, "neovis.js," https://github.com/neo4j-contrib/neovis.js, 2021.

[9] U. Brandes, M. Eiglsperger, J. Lerner, and C. Pich, "Graph markup language (graphml)," in *Handbook of graph drawing visualization*, ser. Discrete mathematics and its applications, R. Tamassia, Ed. CRC Press, 2013, pp. 517–541.

[10] M. Fowler, *Refactoring - Improving the Design of Existing Code*, ser. Addison Wesley object technology series. Addison-Wesley, 1999.

[11] T. Arendt and G. Taentzer, "Uml model smells and model refactorings in early software development phases," Universitat Marburg, Tech. Rep., 2010.

[12] G. Suryanarayana, G. Samarthyam, and T. Sharma, *Refactoring for software design smells: managing technical debt*. Morgan Kaufmann, 2014.

[13] T. Haendler, "On using UML diagrams to identify and assess software design smells," in *Proceedings of the 13th International Conference on Software Technologies*. SciTePress, 2018, pp. 447–455.

[14] H. Mumtaz, M. Alshayeb, S. Mahmood, and M. Niazi, "A survey on uml model smells detection techniques for software refactoring," *Journal of Software: Evolution and Process*, vol. 31, no. 3, p. e2154, 2019.

[15] S. Gérard, C. Dumoulin, P. Tessier, and B. Selic, "Papyrus: A uml2 tool for domain-specific language modeling," in *Dagstuhl Workshop on Model-Based Engineering of Embedded Real-Time Systems*. Springer, 2007, pp. 361–368.

[16] G. Taentzer, K. Ehrig, E. Guerra, J. d. Lara, L. Lengyel, T. Levendovszky, U. Prange, D. Varro, and S. Varro-Gyapay, "Model transformation by graph transformation: A comparative study," 2005.

[17] K. Hölscher, P. Ziemann, and M. Gogolla, "On translating uml models into graph transformation systems," *Journal of Visual Languages & Computing*, vol. 17, no. 1, pp. 78–105, 2006.

[18] S. Buckl, F. Matthes, and C. M. Schweda, "Classifying enterprise architecture analysis approaches," in *IFIP-International Workshop on Enterprise Interoperability*. Springer, 2009, pp. 66–79.

[19] A. Barbosa, A. Santana, S. Hacks, and N. v. Stein, "A taxonomy for enterprise architecture analysis research," in *21st International Conference on Enterprise Information Systems*, vol. 2. SciTePress, 2019, pp. 493–504.

[20] H. A. Proper, D. Bork, and G. Poels, "Towards an Ontology-Driven Approach for Digital Twin Enabled Governed IT Management," in *ODCM-DT'21: First Workshop on Ontology-Driven Conceptual Modelling of Digital Twins*. Springer, 2021, p. in press.

[21] N. Moha, Y.-G. Guéhéneuc, L. Duchien, and A.-F. Le Meur, "Decor: A method for the specification and detection of code and design smells," *IEEE Transactions on Software Engineering*, vol. 36, no. 1, pp. 20–36, 2009.

[22] M. Fokaefs, N. Tsantalis, E. Stroulia, and A. Chatzigeorgiou, "Jdeodorant: identification and application of extract class refactorings," in *33rd International Conference on Software Engineering (ICSE)*. IEEE, 2011, pp. 1037–1039.

Appendix

A short video demonstrating CM2KG Cloud is available at: https://youtu.be/ib763YQ9kCE.