Dynamic Adaption of Metamodels based on Knowledge Graphs

Danial M. Amlashi[1][0000−0003−3628−5725], Alexander Voelz[1][0000−0003−4580−1445], and Junsup Song[2][0000−0002−2167−9222]

[1] Doctoral School Computer Science, University of Vienna, Austria
{danial.mohammadi.amlashi,alexander.voelz}@univie.ac.at
[2] Department of Computer Science and Engineering, Jeonbuk National University, Republic of Korea junsup@jbnu.ac.kr

Abstract. In this work, we report on recent developments regarding the dynamic adaption of metamodels at runtime. This new approach is complemented by *AdoPy*, a Python-based wrapper for metamodel adaption procedures that also facilitates RDF-driven modifications and extensions. The conceptualization and implementation of the approach leverage knowledge graphs to extract relevant classes, relationships, and attributes, enabling the dynamic adaption of modeling method libraries. By integrating these capabilities into ADOxx, the proposed solution links metamodeling and knowledge graphs with systems engineering.

Keywords: Knowledge Graph · Metamodeling · Method Engineering · Semantics-Driven Systems Engineering · AdoPy · ADOxx

1 Introduction

In the field of conceptual modeling and systems engineering, the ability to adapt metamodels dynamically is becoming increasingly relevant for responding to evolving domain-specific requirements. Traditional metamodeling approaches rely on static definitions, hindering flexibility and adaptability during iterative method engineering processes. Knowledge graphs have emerged as a promising solution, offering reasoning capabilities as well as flexible data representation and integration. Associated advancements regarding the integration of semantic data sources focus on enriching and contextualizing modeling-related activities. However, comparable metamodeling-related activities remain an unexplored research field with considerable potential for semantics-driven systems engineering.

To address these challenges, we present the conceptualization and implementation of a dynamic metamodel adaptation approach, which is complemented by *AdoPy*, a Python-based wrapper that facilitates RDF-driven modeling method modifications. This approach leverages knowledge graphs to extract relevant classes, relationships, and attributes, enabling the automated modification of modeling methods through RDF sources. By integrating these capabilities into the ADOxx platform, we validate the proposed solution that aims to bridge knowledge graphs and semantics-driven modeling method engineering.

The remainder of this contribution is structured as follows: Section 2 provides a brief theoretical background by summarizing key characteristics of metamodeling, method engineering, knowledge graphs, and related works. In Section 3, the design problem of this research is formulated. Subsequently, the envisioned integration of knowledge graphs for dynamic modeling method modification and extension within the ADOxx metamodeling platform is outlined in Section 4. Finally, Section 5 discusses our approach's strengths, limitations, and potential future directions before the summarizing conclusion.

2 Theoretical Background

The theoretical foundations of conceptual modeling and knowledge graphs have been covered in previous contributions [30,31,32]. Therefore, this section provides a brief recap of the most relevant aspects, focusing on metamodeling, modeling method engineering, and knowledge graphs. Moreover, related works regarding the integration of knowledge graphs with conceptual modeling and their combined utilization in semantics-driven systems engineering are covered.

2.1 Metamodeling and Modeling Method Engineering

Metamodeling plays a central role in conceptual modeling, aiming to reduce complexity through abstraction and formal representation [25]. Modeling languages provide a structured way to represent systems using well-defined syntax, notation, and semantics, which are governed by metamodels [19]. These metamodels ensure that model instances adhere to a common architecture and enable automatic processing and analysis [22]. According to the Generic Modeling Method Framework (GMMF) [19], a modeling method consists of three key components: the *modeling language*, which specifies how a system is represented with regards to syntax, notation, and semantics; the *modeling procedure*, which guides the practical application of the language; *mechanisms and algorithms*, which enable advanced functionalities, such as model-based code generation.

In case modeling methods are customized for specific application domains, so-called *Domain-Specific Modeling Methods* (DSMMs) are utilized (cf. [20,21]). These methods adapt general modeling concepts to specific domains, providing tailored functionalities that meet the unique requirements of those domains [16]. Moreover, Agile Modeling Method Engineering (AMME) [12] offers a structured framework to develop and refine modeling methods iteratively, ensuring continuous improvement through its life cycle phases: Create, Design, Formalize, Develop, and Deploy. Finally, metamodeling platforms like ADOxx[3] provide the necessary infrastructure for developing, customizing, and deploying DSMMs.

2.2 Knowledge Graphs and Related Works

Knowledge graphs have emerged as a powerful means of representing structured knowledge by explicitly capturing concepts, entities, and their relationships in a

[3] Available at: https://adoxx.org

graph-based format [9]. Although the term lacks an established definition and is often used synonymously with related terms like *Ontology* or *Knowledge Base* [8], it gained widespread recognition after Google introduced its knowledge graph in 2012. In the context of the Semantic Web, knowledge graphs are typically implemented using the Resource Description Framework (RDF), which enables machine-readable, linked representations of data [33]. RDF provides a foundation for Linked Open Data (LOD), facilitating interoperability and seamless data integration across different sources. Beyond static representations, knowledge graphs incorporate reasoning engines that utilize inference for deriving implicit knowledge from explicitly defined relationships [8].

Knowledge Graphs and Conceptual Modeling. An increasing effort has recently been invested in leveraging conceptual modeling through knowledge graph-based capabilities. Before the term knowledge graph was popularized, methods from the ontology field were utilized to enhance traditional conceptual modeling. Initial contributions on *Ontology-Driven Conceptual Modeling* (ODCM) highlight the role of foundational ontologies in addressing semantic interoperability challenges [10]. More recent publications apply ODCM principles to microservice architectures for improved system modularity [24]. Also, the classification of conceptual modeling elements has been automated according to foundational ontology categories through Graph Neural Networks [1]. Moreover, an ontology-based metamodeling approach following the OCDM principles has been proposed, which integrates human- and machine-interpretable models for the purpose of ensuring consistency and adaptability in domain-specific modeling [11].

Another area of research concerns the seamless transformation of conceptual models into knowledge graphs, for which several approaches exist. Early works introduced an RDF serialization functionality in the context of ADOxx [6], which since then has been realized as an extension for direct integration within ADOxx-based modeling methods [3] and utilized for model-driven enterprise data fabric approaches [18]. Following a similar approach, a diagrammatic modeling tool was proposed that enables users to construct RDF graphs visually and generate machine-readable N-triples using metamodeling principles [7]. Considering more platform-agnostic solutions, CM2KG$^{\text{cloud}}$ presents a web-based implementation that transforms conceptual models from different metamodeling platforms (e.g., EMF, ADOxx, Papyrus) into knowledge graphs, supporting formats such as RDF, GraphML, and OWL [27].

Building on the principle of LOD, Linked Open Models (LOM) extend this paradigm by integrating conceptual model information into linked data systems. The resulting LOM approach enables conceptual models to act as structured, human-readable, and semantically rich knowledge representations that enhance the interoperability and query capabilities [14]. Furthermore, the integration of domain-specific models with linked data has been explored to enrich existing linked datasets with additional semantic constraints derived from conceptual modeling [3]. A practical application of LOM is presented in [30], showcasing how conceptual models can be aligned with external knowledge graphs to realize more meaningful knowledge structures through semantic enrichment.

Semantics-Driven Systems Engineering. Knowledge graphs have emerged as key enablers in systems engineering by structuring domain knowledge in a machine-processable format, which enables advanced querying, semantic integration, and reasoning. Early examples include a knowledge-centric management framework that integrates conceptual modeling and knowledge graphs to support service-oriented systems engineering [5] and an interoperability mechanism between ADOxx and GraphDB that enables reasoning over semantic-rich conceptual models [15]. In this context, the notion of *Semantics-Driven Systems Engineering* (SDSE) has recently emerged, emphasizing the transition from traditional model-driven engineering to a knowledge representation-centric approach [2]. An example of applying SDSE principles is provided in the context of smart building management, where an ontology-based metamodeling approach enables engineers to create and maintain domain knowledge through visual models [23]. Moreover, the individual steps underlying SDSE are structured as a repeatable process in [4] that is subdivided into five stages: (i) DSMM conceptualization and realization cycle, (ii) DSMM utilization for knowledge capturing, (iii) knowledge graph generation for various DSMM components, (iv) semantics-driven feature parameterization, and (v) feedback loops for propagating required changes.

3 Problem Statement

The iterative development of DSMMs requires continuous metamodel adaptions to reflect changing domain requirements. Traditionally, metamodeling platforms rely on static metamodel definitions. Consequently, integrating domain-specific changes into modeling method libraries often demands time-intensive manual effort, thus reducing agility. While existing research primarily investigates how conceptual models can enrich or inform knowledge graphs, there remains an unexplored potential in reversing this knowledge flow. Specifically, leveraging RDF-based knowledge graphs to dynamically enrich DSMM metamodels at runtime represents an intriguing direction to address the aforementioned limitations.

This contribution follows the principles of Design Science Research (DSR), focusing on the design, development, and demonstration of a design artifact that addresses a formulated problem statement [34]. Within the DSR methodology, such a problem is systematically defined by clearly distinguishing the artifact, its basic requirements, and associated goals. The core artifact of this research is a dynamic metamodel adaptation approach that integrates RDF-driven knowledge graph modifications into metamodeling environments.

The underlying goal of this contribution is summarized by formulating the design problem according to the DSR template [34]:

> Improve the iterative development of DSMMs (**problem context**)
> ... by introducing a dynamic metamodel adaption approach (**artifact**)
> ... that provides RDF-based information extraction mechanisms as well as integration with a given metamodeling platform (**requirements**)
> ... to facilitate metamodel modifications and extensions at runtime. (**goal**)

4 Dynamic Metamodel Adaption

Metamodels must dynamically adapt to evolving requirements and domain-specific constraints to fully leverage the potential of knowledge graphs in systems engineering. This section introduces an approach for dynamic metamodel adaption, enabling runtime refinements of modeling methods.

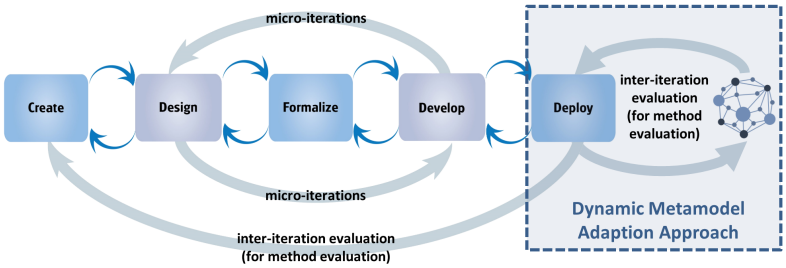
4.1 Conceptualization: Dynamic Metamodel Adaption Approach

The process of modeling method development usually requires change requests to be implemented within the next life cycle iteration. While such an approach ensures consistency, it introduces additional effort when frequent modifications, in the sense of rapid prototyping (cf. [28]), are required. To address this issue, we present a conceptualization of the dynamic metamodel adaption approach that is contextualized within Fig. 1 as an extension to both the AMME life cycle [13] and the high-level view on the repeatable SDSE process [4], respectively.

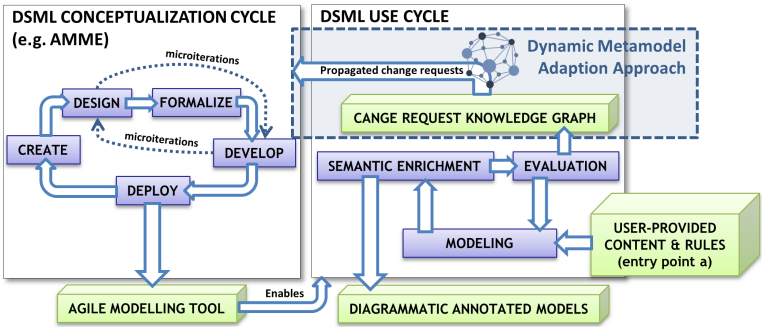
The AMME life cycle, as depicted in Fig 1a, structures the development of DSMMs through five sequential iteration phases. As mentioned previously, adaptions of modeling methods require a complete cycle, incorporating both micro-iterations within phases and inter-iteration evaluations for broader method refinement. The dynamic metamodel adaption approach extends the AMME life cycle by integrating the possibility of knowledge graph-based refinements within the deployment phase, thus not necessitating a new iteration cycle. This mechanism allows structured change requests derived from applying the deployed DSMM in real-world contexts to inform metamodel adaptions dynamically and accelerate method evaluation by reducing manual effort.

Building upon the AMME life cycle extension, Fig. 1b contextualizes the dynamic metamodel adaption approach within the broader SDSE process. More precisely, the first two stages of the SDSE process (cf. Section 2.2) are depicted in Fig. 1b while also considering the fifth stage of propagating required change requests through feedback loops. Subsequently, we assume that the evaluation results associated with the fifth stage can be structured in the form of a *Change Request Knowledge Graph*. Instead of requiring manual adjustments, this graph-based representation enables automated reasoning and structured updates to the metamodel, as described in the previous paragraph. By integrating this knowledge graph-driven feedback loop, the dynamic adaption approach allows for continuous method evolution without necessitating a complete AMME iteration.

Both conceptual perspectives of the dynamic metamodel adaption approach require a dedicated environment to test, validate, and refine developed solutions. The OMiLAB (Open Model Initiative Laboratory) ecosystem provides such an infrastructure, offering a collaborative space for DSMM development and evaluation [17]. With its physical and digital resources, OMiLAB supports the practical application of metamodeling frameworks, enabling researchers and practitioners to experiment with conceptual modeling-based techniques [29]. As our approach relies on structured feedback and dynamic modifications, OMiLAB presents an ideal environment for real-world experimentation and further validation.



(a) Dynamic metamodel adaption within AMME (adapted from [13])



(b) Dynamic metamodel adaption within SDSE (adapted from [4])

Fig. 1. Conceptualization of the dynamic metamodel adaption approach contextualized as an extension within (a) the AMME life cycle and (b) the SDSE process

4.2 Implementation: Dynamic Metamodel Adaptions using *AdoPy*

To realize the conceptual approach outlined in Section 4.1, we implemented *AdoPy*[4], a Python-based wrapper that enables metamodel adaptions for the ADOxx metamodeling platform. A central feature of the *AdoPy* library is the `transform_to_modeltype` function available for the `class KnowledgeGraph`. This function parses a knowledge graph (e.g., in Turtle format) using the `rdflib` library to extract OWL classes (`owl:Class`), their hierarchical relationships (via `rdfs:subClassOf`), object properties (`owl:ObjectProperty`), and datatype properties (`owl:DatatypeProperty`). These are transformed into fitting ADOxx meta^{2}model components such as class instances, their superclasses, relation instances, and attribute instances, with the class hierarchy maintaining inheritance and `_D-construct_` serving as the default root. Moreover, generic graphical notations are included in the transformation process. Finally, the resulting components are exported as ADOxx map (i.e., export.leo) and integrated at runtime as a new model type using the `ASC_GlobalProcedures_MetaModelAtRuntime` script, which provides global procedures for retrieving library information and dynamically extending it with new classes, attributes, and relationships.

[4] Available at: https://adoxx.org/modules/details/?id=487

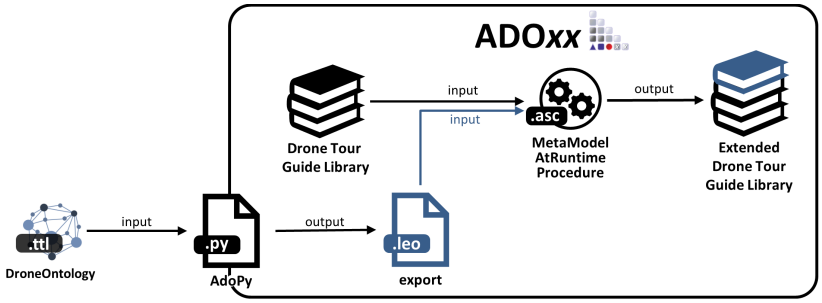


Fig. 2. Dynamic metamodel adaption approach instance: An ADOxx implementation

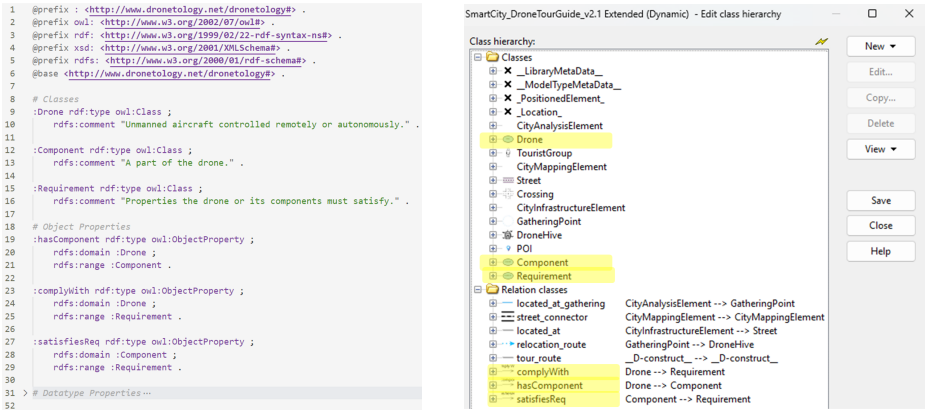
4.3 Validation: Modeling Method Library Configuration in ADOxx

For the validation of the dynamic metamodel adaptation approach, we extend an existing ADOxx-based modeling method that is associated with the drone tour guide scenario presented in [30]. In the contribution, semantic matching was utilized to enrich city tour models of a DSMM with LOD from DBpedia and Wikidata. To validate the conceptualization and implementation of our proposed approach, we assume that the DSMM has been deployed, tested, and evaluated. Moreover, it is implied that the resulting change requests include the integration of a new model type named *Drone Specification*, which provides the possibility to detail each drone based on its components and associated requirements. Lastly, the model type specification is assumed to be available in form of a Turtle file, which we derived from the Dronetology[5] by manually extracting suitable elements. Based on these assumptions, Fig. 2 displays an instance of the dynamic metamodel adaption approach implemented in the context of ADOxx.

The exemplary validation process in Fig. 2 starts by providing the extracted *DroneOntology*, which is subsequently displayed in Fig. 3a, as an input to *AdoPy*. Next, the `transform_to_modeltype` function is utilized to transform relevant classes and properties into corresponding meta^{2}model components, which are compiled in a model type and exported as ADOxx-specific map (cf. Section 4.2). Finally, the *export* file is used to extend the existing modeling method library via *MetaModelAtRuntime* procedures. The resulting extensions are highlighted in Fig. 3b, which displays the class hierarchy of the extended library.

The integrated *Drone Specification* model type enables users to create drone models within the ADOxx environment that are dedicated to capturing relevant components and requirements. More specifically, drone instances can be linked to selected components and requirements through corresponding relations. In addition, a given component can also be associated with certain requirements to express that it satisfies these requirements. In this way, the semantic structure and hierarchy of the *DroneOntology* are preserved, as detailed in the displayed comparison between the Turtle-based input file and the resulting extension to the class hierarchy of the drone tour guide library (cf. Fig. 3).

[5] Available at: https://www.dronetology.net/dronetology/index-en.html



(a) DroneOntology.ttl file (b) Extended drone tour guide library

Fig. 3. Comparison of (a) the Turtle-based input file and (b) the resulting extension to the class hierarchy of the drone tour guide library in ADOxx (cf. Fig. 2)

This validation example showcases that *AdoPy* effectively extends modeling methods at runtime for the purpose of modeling drones and their requirements, thus demonstrating its potential for broader applications in the SDSE context.

5 Discussion

This research introduces a metamodel adaption approach that enhances SDSE by leveraging knowledge graph-based feedback to dynamically modify and extend already deployed modeling methods. While the proposed approach enables a more flexible way of addressing change requests that result from the evaluation phase, several limitations must be discussed to ensure its broader applicability.

In the context of the applied DSR methodology (cf. Section 3), OMiLAB has served as an experimentation environment that provides the necessary infrastructure for testing and validating developed artifacts, especially within the demonstration phase. However, current limitations have motivated a reiteration back to the *Design and Development* phase (cf. [26]) to enhance mechanisms of the `ASC_GlobalProcedures_MetaModelAtRuntime` extension[6], which underpin the core capabilities of our approach. Future research will focus on a broader integration within SDSE processes by addressing current limitations. Such limitations and related prospects are outlined in the remainder of the discussion.

A primary constraint of our approach is the manual steps currently required to extend a modeling method library. As a reference, each transformation step illustrated in Fig. 2 must be manually executed. This includes the initial import of the *DroneOntology* within *AdoPy*, the subsequent execution of

[6] Ongoing developments are available at: https://adoxx.org/modules/details/?id=188

the `transform_to_modeltype` function, and the concluding utilization of the *MetaModelAtRuntime* procedure, which takes the generated *export* file and the existing library as inputs to output the extended library. In addition, the DSMM deployment, testing, and evaluation are prerequisites, meaning that a modeling method library must already exist before it can be dynamically refined. This further implies that the dynamic adaptation process is not fully automated, as human intervention is necessary for developing the DSMM in the first place and at multiple subsequent stages. Achieving higher levels of automation remains a critical area of improvement for the future.

Another limitation relates to the platform dependency of our approach. The current implementation is tightly coupled with ADOxx, as it relies on *AdoPy* for processing knowledge graph-driven metamodel extensions and on procedures provided by the `ASC_GlobalProcedures_MetaModelAtRuntime` script. While ADOxx offers robust support for metamodeling, extending this approach to other platforms requires additional effort. Moreover, the assumption that derived change requests can be automatically structured into a knowledge graph poses challenges, as feedback data from real-world modeling tool applications is often unstructured and heterogeneous. Transforming this feedback into a structured knowledge graph representation usually remains a human effort that demands more sophisticated semantic processing techniques.

Despite these limitations, the proposed approach for dynamic metamodel adaptions represents a meaningful step toward flexible and efficient modeling method development in the sense of rapid prototyping. By continuously evolving metamodels based on structured feedback and reasoning, this work contributes to the ongoing evolution of SDSE. Beyond dynamic refinements, our approach also opens up opportunities for automated library initialization, enabling the creation of an initial DSMM structure directly from knowledge graphs. This could significantly reduce the manual effort required for early-stage DSMM development, thereby addressing one limitation of the current approach. By extending this mechanism, future work will explore how entire modeling environments can be dynamically instantiated and evolved, further reinforcing the role of rapid, knowledge-driven prototyping within SDSE. In this context, OMiLAB provides the ecosystem to support these advancements, ensuring a practical foundation for implementing and refining both current and upcoming developments.

Moreover, this research contributes to the scientific field by reversing the traditional knowledge-flow direction between DSMMs and knowledge graphs. Whereas prior research primarily investigates how metamodel-based designs can enhance knowledge graphs (cf. Section 2.2), this approach leverages RDF-based inputs to dynamically adapt metamodels at runtime. This reverse flow represents a promising new perspective, expanding the capabilities and agility of SDSE.

6 Conclusion

In the field of conceptual modeling and SDSE, the ability to adapt metamodels dynamically offers considerable advantages for responding to constantly evolving

requirements. This work introduces an approach to metamodel adaptations at runtime, leveraging knowledge graphs to enable the dynamic extension of deployed modeling methods. The proposed approach utilizes *AdoPy*, which allows for RDF-driven modification and extension of ADOxx libraries. Such procedures are especially relevant within SDSE process stages, where feedback mechanisms guide the evaluation of DSMMs. Instead of treating generated feedback as static input for the next AMME iteration cycle, we assume a knowledge graph-based specification of change requests that can be leveraged to enhance agility and reduce manual effort in iterative DSMM development by utilizing the dynamic metamodel adaption approach proposed in this work. The corresponding validation demonstrates our proposed approach's feasibility and practical utility in the context of a drone tour guide case. This establishes a new perspective in SDSE, where modeling methods continuously evolve through structured, semantics-driven feedback rather than discrete iteration cycles. Future works will focus on refining the metamodel adaption approach, for example, by automating the modeling method extension process with the help of LOD-based suggestions, and on addressing the remaining limitations discussed in this work.

References

1. Ali, S.J., Guizzardi, G., Bork, D.: Enabling representation learning in ontology-driven conceptual modeling using graph neural networks. In: Advanced Information Systems Engineering, pp. 278–294. Springer Nature Switzerland (2023). https://doi.org/10.1007/978-3-031-34560-9_17
2. Buchmann, R., Eder, J., Fill, H.G., Frank, U., Karagiannis, D., Laurenzi, E., Mylopoulos, J., Plexousakis, D., Santos, M.Y.: Large language models: Expectations for semantics-driven systems engineering. Data & Knowledge Engineering **152**, 102324 (2024). https://doi.org/10.1016/j.datak.2024.102324
3. Buchmann, R.A., Karagiannis, D.: Enriching linked data with semantics from domain-specific diagrammatic models. Business & Information Systems Engineering **5**(58), 341–353 (2016). https://doi.org/10.1007/s12599-016-0445-1
4. Buchmann, R.A.: Semantics-driven systems engineering: Requirements and prerequisites for a new flavor of model-driven engineering. In: Metamodeling: Applications and Trajectories to the Future, pp. 19–34. Springer Nature Switzerland (2024). https://doi.org/10.1007/978-3-031-56862-6_2
5. Buchmann, R.A., Ghiran, A.M.: Serviceology-as-a-service: a knowledge-centric interpretation. In: Serviceology for Services, pp. 190–201. Springer International Publishing (2017). https://doi.org/10.1007/978-3-319-61240-9_18
6. Buchmann, R.A., Karagiannis, D.: Agile modelling method engineering: Lessons learned in the comvantage research project. In: The Practice of Enterprise Modeling, pp. 356–373. Springer International Publishing (2015). https://doi.org/10.1007/978-3-319-25897-3_23
7. Chis, A., Buchmann, R.A., Ghiran, A.M.: Towards a modeling method for low-code knowledge graph building. In: Proceedings of the 16th IFIP WG 8.1 Working Conference on the Practice of Enterprise Modeling and the 13th Enterprise Design and Engineering Working Conference (PoEM & EDEWC - Companion 2023). CEUR-WS (2023), https://ceur-ws.org/Vol-3645/forum4.pdf

8. Ehrlinger, L., Wöß, W.: Towards a definition of knowledge graphs. In: Martin, M., Cuquet, M., Folmer, E. (eds.) Joint Proceedings of the 1st International Workshop on Semantic Change & Evolving Semantics (SuCCESS'16) co-located with the 12th International Conference on Semantic Systems (SEMANTiCS 2016). vol. 1695. CEUR-WS.org (2016), https://ceur-ws.org/Vol-1695/paper4.pdf
9. Flasiński, M.: Symbolic artificial intelligence. In: Introduction to Artificial Intelligence, pp. 15–22. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-40022-8_2
10. Guizzardi, G.: The role of foundational ontologies for conceptual modeling and domain ontology representation. In: 2006 7th International Baltic Conference on Databases and Information Systems. pp. 17–25. IEEE (2006). https://doi.org/10.1109/dbis.2006.1678468
11. Hinkelmann, K., Laurenzi, E., Martin, A., Thönssen, B.: Ontology-based metamodeling. In: Business Information Systems and Technology 4.0, pp. 177–194. Springer International Publishing (2018). https://doi.org/10.1007/978-3-319-74322-6_12
12. Karagiannis, D.: Agile modeling method engineering. In: Karanikolas, N., Akoumianakis, D., Nikolaidou, M., Vergados, D., Xeno, M. (eds.) Proceedings of the 19th Panhellenic Conference on Informatics. p. 5–10. Association for Computing Machinery, New York, NY, USA (2015). https://doi.org/10.1145/2801948.2802040
13. Karagiannis, D.: Conceptual modelling methods: The amme agile engineering approach. In: Karagiannis, D., Lee, M., Hinkelmann, K., Utz, W. (eds.) Domain-Specific Conceptual Modeling: Concepts, Methods and ADOxx Tools, pp. 3–21. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-030-93547-4_1
14. Karagiannis, D., Buchmann, R.A.: Linked open models: Extending linked open data with conceptual model information. Information Systems **56**, 174–197 (2016). https://doi.org/10.1016/j.is.2015.10.001
15. Karagiannis, D., Buchmann, R.A.: A proposal for deploying hybrid knowledge bases: the adoxx-to-graphdb interoperability case. In: Proceedings of the 51st Hawaii International Conference on System Sciences. pp. 4055–4064 (2018). https://doi.org/10125/50399
16. Karagiannis, D., Buchmann, R.A., Burzynski, P., Reimer, U., Walch, M.: Fundamental conceptual modeling languages in omilab. In: Karagiannis, D., Mayr, H.C., Mylopoulos, J. (eds.) Domain-Specific Conceptual Modeling: Concepts, Methods and Tools, pp. 3–30. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-39417-6_1
17. Karagiannis, D., Buchmann, R.A., Utz, W.: The omilab digital innovation environment: Agile conceptual models to bridge business value with digital and physical twins for product-service systems development. Computers in Industry **138**, 103631 (2022). https://doi.org/10.1016/j.compind.2022.103631
18. Karagiannis, D., Burzynski, P., Utz, W., Buchmann, R.A.: A metamodeling approach to support the engineering of modeling method requirements. In: 2019 IEEE 27th International Requirements Engineering Conference (RE). pp. 199–210. IEEE (2019). https://doi.org/10.1109/re.2019.00030
19. Karagiannis, D., Kühn, H.: Metamodelling platforms. In: Bauknecht, K., Tjoa, A.M., Quirchmayr, G. (eds.) E-Commerce and Web Technologies. p. 182. Springer, Berlin, Heidelberg (2002). https://doi.org/10.1007/3-540-45705-4_19
20. Karagiannis, D., Lee, M., Hinkelmann, K., Utz, W. (eds.): Domain-Specific Conceptual Modeling: Concepts, Methods and ADOxx Tools. Springer International Publishing, Cham, 1 edn. (2022). https://doi.org/10.1007/978-3-030-93547-4

21. Karagiannis, D., Mayr, H.C., Mylopoulos, J. (eds.): Domain-Specific Conceptual Modeling: Concepts, Methods and Tools. Springer International Publishing, Cham, 1 edn. (2016). https://doi.org/10.1007/978-3-319-39417-6
22. Kühn, H., Junginger, S., Karagiannis, D., Petersen, C.: Metamodellierung im geschäftsprozeßmanagement: Konzepte, erfahrungen und potentiale. In: Desel, J., Pohl, K., Schürr, A. (eds.) Teubner Reihe Wirtschaftsinformatik, vol. 12923, pp. 75–90. Vieweg+Teubner Verlag, Wiesbaden (1999). https://doi.org/10.1007/978-3-322-93104-7_5
23. Laurenzi, E., Allan, J., Campos, N., Stoller, S.: An ontology-based meta-modelling approach for semantic-driven building management systems. In: Advanced Information Systems Engineering Workshops, pp. 200–211. Springer Nature Switzerland (2024). https://doi.org/10.1007/978-3-031-61003-5_18
24. Morais, G., Bork, D., Adda, M.: Towards an ontology-driven approach to model and analyze microservices architectures. In: Proceedings of the 13th International Conference on Management of Digital EcoSystems. pp. 79–86. MEDES '21, ACM (2021). https://doi.org/10.1145/3444757.3485108
25. Mylopoulos, J.: Conceptual modelling and telos. In: P., L., R., Z. (eds.) Conceptual modelling, databases, and CASE: An integrated view of information system development, pp. 49–68. John Wiley & Sons, New York (1992)
26. Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S.: A design science research methodology for information systems research. Journal of Management Information Systems **24**(3), 45–77 (2007). https://doi.org/10.2753/mis0742-1222240302
27. Smajevic, M., Ali, S.J., Bork, D.: Cm2kgcloud – an open web-based platform to transform conceptual models into knowledge graphs. Science of Computer Programming **231**, 103007 (2024). https://doi.org/10.1016/j.scico.2023.103007
28. Tripp, S.D., Bichelmeyer, B.: Rapid prototyping: An alternative instructional design strategy. Educational Technology Research and Development **38**(1), 31–44 (1990). https://doi.org/10.1007/bf02298246
29. Vaidian, I., Jurczuk, A., Misiak, Z., Neidow, M., Petry, M., Nemetz, M.: Challenging digital innovation through the omilab community of practice. In: Karagiannis, D., Lee, M., Hinkelmann, K., Utz, W. (eds.) Domain-Specific Conceptual Modeling: Concepts, Methods and ADOxx Tools, pp. 41–64. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-030-93547-4_3
30. Voelz, A., Amlashi, D.M., Lee, M.: Semantic matching through knowledge graphs: A smart city case. In: Ruiz, M., Soffer, P. (eds.) Advanced Information Systems Engineering Workshops. pp. 92–104. Springer International Publishing, Cham (2023). https://doi.org/10.1007/978-3-031-34985-0_10
31. Völz, A., Amlashi, D.M., Burzynski, P., Utz, W.: Adoxx: Eine low-code-plattform für die entwicklung von modellierungswerkzeugen. HMD Praxis der Wirtschaftsinformatik **61**(5), 1295–1316 (2024). https://doi.org/10.1365/s40702-024-01096-x
32. Völz, A., Vaidian, I.: Digital transformation through conceptual modeling: The nemo summer school use case. In: Modellierung 2024, pp. 139–156. Gesellschaft für Informatik e.V., Bonn (2024). https://doi.org/10.18420/modellierung2024_014
33. W3C: Rdf - resource description framework (2014), https://www.w3.org/RDF/, (accessed 20 February 2025)
34. Wieringa, R.J.: Design Science Methodology for Information Systems and Software Engineering. Springer Berlin Heidelberg (2014). https://doi.org/10.1007/978-3-662-43839-8