Federation of heterogeneous models with machine learning-assisted model views

James William Pontes Miranda

▶ **To cite this version:**

James William Pontes Miranda. Federation of heterogeneous models with machine learning-assisted model views. Computer Science [cs]. Ecole nationale supérieure Mines-Télécom Atlantique, 2025. English. NNT : 2025IMTA0454 . tel-04934150

HAL Id: tel-04934150

https://theses.hal.science/tel-04934150v1

Submitted on 7 Feb 2025

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

COLLEGES SCIENCES
BRETAGNE POUR L'INGENIEUR
LOIRE ET LE NUMERIQUE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPÉRIEURE
MINES-TÉLÉCOM ATLANTIQUE BRETAGNE
PAYS DE LA LOIRE – IMT ATLANTIQUE

ÉCOLE DOCTORALE N° 648
Sciences pour l'Ingénieur et le Numérique
Spécialité : *Informatique*

Par

James William PONTES MIRANDA

Federation of Heterogeneous Models with Machine Learning-Assisted Model Views

Thèse présentée et soutenue à IMT Atlantique, Nantes, le 24 Janvier 2025
Unité de recherche : Laboratoire des Sciences du Numérique de Nantes (LS2N) - (UMR 6004)
Thèse N° : 2025IMTA0454

Rapporteurs avant soutenance :

Yves LEDRU	Professor, Université Grenoble Alpes
Houari SAHRAOUI	Professor, Université de Montréal

Composition du Jury :

Président :	Dániel VARRÓ	Professor, Linköping University
Examinateurs :	Yves LEDRU	Professor, Université Grenoble Alpes
	Houari SAHRAOUI	Professor, Université de Montréal
	Catia TRUBIANI	Associate Professor, Gran Sasso Science Institute
Dir. de thèse :	Massimo TISI	Professor, IMT Atlantique
Co-encadrant de thèse :	Hugo BRUNELIERE	Researcher, IMT Atlantique
	Gerson SUNYÉ	Associate Professor, HDR, University of Nantes

Acknowledgement

This thesis reached its final milestone thanks to the support of many people. I would not be the same person, nor would I be here today, without their support.

I must begin by thanking my mother, who did everything she could to support my education, and my entire family, who have encouraged me throughout this long journey since the start of my PhD.

I appreciate the encouragement of my former professors and supervisors, who recognized my potential for academic work a long time ago and motivated me to have higher goals. To my former manager, Diego Matos, I am incredibly grateful for your mentorship and kindness during my career transitions, especially as I adapted to life in France.

I want to express my gratitude to my dear supervisors Massimo Tisi, Hugo Bruneliere, and Gerson Sunyé. Your guidance, encouragement, and patience have shaped me as a researcher, and I feel immensely fortunate to have had you as mentors.

I am also thankful to all NaoModers, especially my office mates who were always positively incentivized to keep going with their words or attitudes during these years: Jolan, Josselin, Ali, Yasmina, and Zak.

I wish to acknowledge the AIDOaRt project for its support and the opportunities it provided to collaborate with brilliant and dedicated individuals. I especially thank those associated with the VCE use case and the members of the French cluster for their collaboration and insights. Your contributions have been integral to the success of this work.

Many thanks to the jury members, Daniel Varró, Catia Trubiani, Houari Sahraoui, and Yves Ledru, for their respectful and thoughtful feedback which brought up many new ideas. Your input has enriched this work and sparked new ideas for the future.

Finally, my wife Lana deserves far more than a simple thank you on a page of my thesis. My life changed and keeps changing for the better thanks to you during the last decade on your side. Thank you for being my partner in this PhD journey and in all the ones to come. You are the source of my optimism. Because of you, I see a bright future ahead, and I know it's worth it because I get to share it with you.

This work is dedicated to Lana, for inspiring me every day.

I have always wished for my computer to be as easy to use as my telephone; my wish has come true because I can no longer figure out how to use my telephone.

—Bjarne Stroustrup

RÉSUMÉ LONG EN FRANÇAIS

Contexte Général

Les transformations numériques façonnent le monde. Cela est particulièrement évident lorsque l'on examine l'évolution des systèmes Internet des objets (IdO) [1] et la croissance des systèmes numériques très complexes, tels que les Système Cyber-Physiques (SCPs) [2]. Essentiellement, les SCPs intègrent des composants logiciels et des artefacts associés avec les éléments physiques en utilisant un ensemble de capteurs et d'actionneurs qui doivent communiquer constamment entre eux de manière contextuelle. Avec l'avènement de l'*Industrie 4.0*, les SCPs modernes ont évolué pour non seulement collecter et gérer des données, mais aussi pour soutenir des mécanismes de traçabilité qui facilitent la surveillance en temps réel et la traçabilité des données [2-4]. Cette traçabilité vise à faciliter la récupération d'informations pertinentes à différents stades du traitement des données et des opérations système [3, 4]. On trouve des exemples de SCPs dans différents domaines, par exemple les systèmes de contrôle industriels [5], les entreprises intelligentes [6], les soins de santé [7, 8], et les systèmes avioniques [9].

L'ingénierie des SCPs est souvent difficile car la capacité à automatiser efficacement les activités dépend principalement de la qualité et de la pertinence des données et outils disponibles [10]. Développer des SCPs à partir de zéro est un défi, car les ingénieurs doivent gérer plusieurs problèmes, notamment l'hétérogénéité des données, un environnement multi-acteurs, et des contraintes de concurrence ou de temps, par exemple [11]. Un défi clé de ce type de système que nous voulons souligner est la fragmentation de l'information parmi les différents besoins des parties prenantes, en fonction de leurs profils et niveaux d'expertise [11, 12]. Des outils adéquats supportant la traçabilité des données et la fédération sont nécessaires pour améliorer les processus d'ingénierie des SCPs et surmonter ce défi.

Le Ingénierie Dirigée par les Modèles (IDM) est une approche d'ingénierie où les modèles sont utilisés comme artefact principal, ce qui peut aider à favoriser la réutilisation des connaissances et des données dans l'automatisation des processus [13]. Il permet différentes représentations du même système (ou d'une partie du système) via des abstractions

adaptées avec des techniques et des outils pour manipuler ces abstractions [13-15]. Le IDM offre des moyens appropriés pour relever les défis de l'ingénierie des SCPs mentionnés précédemment [10, 15-18]. Cependant, certaines limitations subsistent. Parmi les exemples, on trouve les abstractions adaptées pour modéliser les composants physiques en abstractions informatiques [10], la conception d'outils de simulation spécifiques au domaine [15] et le support d'outils pour la fédération automatisée [19] La thèse présentée met en avant le défi de fédérer plusieurs modèles ou points de vue en un ensemble cohérent, ce qui nécessite un support d'outils sophistiqués [19, 20]. La fédération automatisée aide à intégrer des modèles disparates, assurant la cohérence et l'interopérabilité, ce qui est crucial pour les projets à grande échelle et multi-disciplinaires, mais reste un domaine avec un support limité [19, 21].

L'Intelligence Artificielle (IA) apporte un ensemble d'algorithmes et de méthodes pour automatiser des solutions à des problèmes complexes par des prédictions, des informations basées sur les données et des recommandations [22-24]. Les dernières avancées en IA impactent tous les aspects du développement des systèmes, de la spécification à la maintenance, affectant leur conception, test et déploiement. Parmi tous les sous-domaines de l'IA, ce travail s'intéresse au Apprentissage Automatique (AA) et, plus précisément, aux techniques de Apprentissage Profond (AP) [25], qui utilisent des Réseau de Neurones Artificielss (RNAs) pour apprendre des modèles complexes et prédire des scénarios inconnus. Cet intérêt particulier est dû au fait que, parmi toutes les techniques étudiées sous la bannière de l'IA, les percées récentes sont très souvent soutenues par une certaine variation de AP, même lorsqu'elles sont combinées à une autre technique de AA [26, 27]. Cette expansion de l'IA contribue également au développement des SCPs, aidant à résoudre des problèmes liés à la prise de décision et permettant des prédictions plus précises car l'IA peut analyser un volume de données généré en faible latence [24]. Dans ce contexte, le AP peut être utilisé pour améliorer les techniques traditionnelles de IDM et ainsi améliorer le support actuel pour l'ingénierie des SCPs en fournissant les outils nécessaires pour les défis restants.

Ces dernières années, en réponse aux défis énoncés, divers partenariats industrie-université ont été mis en place pour améliorer la qualité du développement, de l'intégration et de la maintenance de l'ingénierie des SCPs, e. g. le projet MegaM@Rt2 [28] [1] et le projet AIDOaRt [29]. [2] Le premier portait sur l'utilisation du IDM pour gérer l'in-

1. `https://megamart2-ecsel.eu/` *(Last Accessed in November 2024)*
2. `https://www.aidoart.eu/` *(Last Accessed in November 2024)*

tégration des aspects de conception et d'exécution d'un système donné, démontrant les capacités d'utilisation du IDM dans le processus d'ingénierie des SCPs et identifiant également certaines des limitations mentionnées ci-dessus [30-32]. L'AIDOaRt est le projet dans lequel les principales contributions de cette thèse ont été développées. Il propose de construire un framework basé sur des modèles pour analyser les données d'exécution et de conception afin de trouver des solutions augmentées par l'IA. Aligné avec nos objectifs, nous pouvons trouver des exemples d'autres projets réussis traitant des projets industriels utilisant à la fois le IDM (par exemple iDev40 [33] [3]) et l'IA (par exemple AI4DI [34] [4]), ce qui montre qu'il est possible de combiner les deux, en vue de résoudre différents défis pour l'ingénierie des SCPs.

Cette thèse vise à montrer une voie possible pour combiner IDM et AP, en abordant un point sensible du développement des SCPs : comment assister les ingénieurs dans la combinaison de différents modèles (i. e. différentes vues du même système) au cours des différentes activités du processus d'ingénierie des SCPs. Nous souhaitons aider les modélisateurs à combiner efficacement les modèles en *vues de modèle*, permettant ainsi à un large éventail de parties prenantes de participer efficacement au développement et à l'utilisation du système. Plus précisément, nous souhaitons démontrer la faisabilité de tirer parti de la puissance de l'IA pour améliorer les activités de modélisation, en particulier la création de vues de modèle. Ce chapitre introductif fournit les connaissances de haut niveau nécessaires sur IDM et IA (à savoir AP en tant que sous-domaine de AA). La section suivante montre comment nous proposons d'aborder ce problème, en présentant la méthodologie ainsi qu'une vue d'ensemble de nos deux principales contributions.

Ingénierie Dirigée par les Modèles et Vues sur le Modèle

Créer des modèles pour faciliter la compréhension de différents systèmes est au cœur de nombreux domaines scientifiques. L'idée est d'utiliser des abstractions pour représenter un système (ou une partie de celui-ci) de manière à ce que la représentation (i. e. le modèle) réponde aux questions à la place du système réel [35]. Dans cette thèse, nous nous intéressons particulièrement à la modélisation comme une partie essentielle de l'ingénierie des systèmes et des logiciels, notamment dans les systèmes complexes et multi-acteurs, comme c'est le cas pour les SCPs.

IDM est une approche qui vise à appliquer des méthodologies et des outils de modé-

3. `https://www.idev40.eu/` *(Last Accessed in November 2024)*
4. `https://ai4di.eu/` *(Last Accessed in November 2024)*

lisation à l'ingénierie des systèmes, en utilisant les modèles comme artefacts de premier ordre [13]. Dans le contexte de la IDM, un *système* est un artefact du monde réel (par exemple un système complexe comme un SCP) que les modèles peuvent représenter. Les *modèles* sont des représentations abstraites d'un système utilisées pour planifier, analyser, investiguer, et parfois générer (des parties du) le système. Pour créer ces modèles, ils doivent suivre la structure et les règles établies par un *métamodèle* donné. Ces métamodèles définissent des *langages de modélisation* spécifiques permettant aux utilisateurs de créer des modèles selon une syntaxe et une sémantique bien définies. Ces langages sont souvent appelés Langage Dédiés (LDs) car ils sont adaptés à des domaines d'application ou à des types de systèmes particuliers. Il est possible de convertir un modèle en un autre en suivant un ensemble de règles exprimées dans une *transformation de modèle*. Les transformations sont utiles pour affiner, traduire, rapporter ou analyser le système modélisé.

Bien que des systèmes plus simples puissent être représentés par des modèles simples, les systèmes complexes nécessitent souvent des modèles plus nombreux et plus complexes, souvent écrits avec différents langages de modélisation (i. e. métamodèles). Chaque langage de modélisation a émergé pour répondre à des besoins spécifiques dans différents domaines. Cette prolifération a augmenté l'hétérogénéité au sein de l'écosystème de modélisation, créant des défis pour l'interopérabilité et l'intégration [36], ce qui a conduit à des situations exigeant différentes stratégies de *gestion des (méta)modèles* [20, 37]. Un défi majeur de gestion se pose lorsque différents acteurs avec divers niveaux d'expertise et un large éventail de besoins créent, gèrent et utilisent des modèles écrits dans plusieurs langages à des fins spécifiques [21]. Différentes stratégies ont été proposées pour gérer cette *hétérogénéité des modèles*, par exemple en utilisant des transformations de modèles pour réunir tous les métamodèles sous un même paradigme opérationnel [38], la composition de (méta)modèles [39], et l'unification via des langages intermédiaires [40].

Une façon de relever ce défi est d'utiliser ce que l'on appelle la *fédération de modèles* [36, 41], ce qui signifie que les modèles développés par différentes équipes ou parties prenantes doivent être liés entre eux pour offrir un mécanisme d'intégration cohésive de modèles hétérogènes [36, 42].

Plusieurs stratégies permettent de réaliser la fédération de modèles en IDM. Ces techniques visent généralement à faciliter l'intégration, la synchronisation et la gestion des modèles issus de différents domaines ou outils, tout en maintenant leur indépendance. La fusion de modèles, le méga-modélisme, la modélisation collaborative et le tissage de

modèles en sont quelques exemples. La fusion de modèles combine deux ou plusieurs modèles en un seul modèle, en résolvant les conflits et incohérences [43, 44]. Le méga-modélisme gère des collections de modèles, définissant les relations sémantiques entre eux et conservant une description de haut niveau de l'interaction des modèles sans en modifier les structures internes [28, 45-47]. La modélisation collaborative soutient la fédération de modèles en permettant à plusieurs parties prenantes de collaborer sur des modèles via des représentations partagées et des mécanismes pour synchroniser les changements dans des environnements distribués [48-50]. Le tissage de modèles consiste à intégrer plusieurs modèles ou métamodèles en une représentation unifiée en établissant des liens explicites entre leurs éléments, permettant la modularité et la réutilisabilité des modèles [51-54]. Ces techniques peuvent être combinées entre elles pour atteindre une fédération de modèles efficace [53, 55] et également avec d'autres techniques non-IDM, comme la combinaison avec DevOps [54], par exemple. Il convient également de mentionner que la fédération de modèles est l'objet d'une norme industrielle par le biais de l'"ISO/TC 184/SC 5" [56].

Enfin, nous avons les vues de modèle comme une autre stratégie pour gérer la fédération de modèles. Essentiellement, cette approche vise à relever le défi de la fragmentation de l'information en ne présentant que les informations pertinentes à chaque partie prenante [21]. Une *model view* est un artefact de modélisation unique (i. e. une représentation d'un système) composé d'éléments provenant de différents modèles. Eventuellement, elles sont complétées par des interconnexions entre eux et des données additionnelles, soit saisies manuellement, soit calculées automatiquement [42, 57]. En d'autres termes, une model view représente le système sous une perspective spécifique donnée par un point de vue [21, 57, 58].

À cet égard, différentes propositions ont émergé dans la communauté de la modélisation pour implémenter le concept de fédération de modèles à travers des vues de modèle [19, 21], mettant effectivement en œuvre les idées du paradigme de *modélisation multi-vues des logiciels et des systèmes* [19, 59, 60]. Pour le cadre de cette thèse, nous nous intéressons particulièrement à la modélisation multi-vues par des solutions basées sur l'utilisation de vues de modèle. Nous restreignons notre analyse aux solutions qui incluent potentiellement (i) un ou des langage(s) dédié(s) à la description de vues (points de vue) et/ou (ii) des mécanismes de virtualisation pour les relations inter-modèles. Pour illustrer avec quelques exemples, nous pouvons citer OpenFlexo [61], [5] VIATRA [62, 63], [6]

5. `https://openflexo.org/` *(Last Accessed in November 2024)*
6. `https://eclipse.dev/viatra/` *(Last Accessed in November 2024)*

et EMF Views [64]. [7] Les détails de ces solutions et d'autres complémentaires sont fournis dans la section 4.5.1.

La technologie des vues de modèle et son application sont au cœur de cette thèse, et nous utilisons *EMF Views* comme l'outil de choix pour étudier et prototyper nos contributions proposées. EMF Views a été développé au sein de notre groupe de recherche (Naomod [8]) ces dernières années [42, 64, 65], et c'est l'outil utilisé dans le cadre du projet AIDOaRt, ce qui souligne sa pertinence pratique pour l'ingénierie des SCPs. EMF Views a déjà montré sa faisabilité dans différentes applications industrielles telles que l'intégration de composants critiques pour des systèmes logiciels [65] et l'ingénierie d'équipements de construction [66], par exemple. Nous fournissons tous les détails nécessaires de cet outil dans la section 2.1.3. EMF Views a été construit avec une expressivité explicite, inspirée par les vues de base de données et des mécanismes non intrusifs qui ne nécessitent pas de modifications des modèles de base, appliquant effectivement des mécanismes de virtualisation. Une analyse comparative entre les fonctionnalités d'EMF Views et d'autres solutions de vues de modèle est fournie dans la section 4.5.1.

Intelligence Artificielle et Ingénierie Dirigée par les Modèles

Dans le contexte de cette thèse, l'objectif principal de l'IA est d'aider les ingénieurs à produire des systèmes plus rapidement et avec une qualité améliorée, tout en traitant des problèmes de plus en plus complexes [67]. Pour ce faire, elle utilise principalement des techniques de AP [25]. Cela s'applique à de nombreuses applications, notamment la vision, la reconnaissance et la génération de parole, le traitement du langage naturel, la génération d'images et de vidéos, les systèmes multi-agents, la planification, la prise de décision et l'intégration de la vision et du contrôle moteur pour la robotique [67]. Elle est également devenue la technique *de facto* utilisée pour l'ingénierie logicielle [68].

Dans le contexte de la IDM, une liste non exhaustive d'applications de AA inclut la réparation de modèles [69], la classification automatique de référentiels de métamodèles [70] et l'extraction automatique de besoins pour une utilisation dans le contexte de la IDM [71]. En effet, l'utilisation correcte de AA est un défi identifié pour les prochaines étapes de l'évolution de la IDM [20].

Barriga et al. [72] ont identifié un large éventail de techniques de AA pour traiter spécifiquement les activités de réparation de modèles, allant des méthodes à règles [73] aux

7. `https://www.atlanmod.org/emfviews/` *(Last Accessed in November 2024)*
8. `https://naomod.github.io/` *(Last Accessed in November 2024)*

arbres de décision [74], et incluant également l'utilisation de AP via des RNAs [75]. Selon les auteurs, bien que les méthodes basées sur des règles offrent des solutions adaptées à des contraintes spécifiques en réparation de modèles (par exemple résolution de violations de cardinalité) [72, 73], les arbres de décision équilibrent simplicité et efficacité, ce qui les rend adaptés à de nombreux scénarios de réparation automatisée [72, 74, 76]. Dans le domaine du AP, l'utilisation des RNAs permet de gérer des dépendances complexes dans les éléments de modèle, améliorant ainsi l'adaptabilité et la scalabilité des solutions pour la réparation de modèles.

Une architecture latente de RNA pour les problèmes de modélisation est l'utilisation des Graph Neural Networks (GNNs) [77]. Elle s'aligne bien avec les exigences de IDM puisque les graphes étiquetés sont des structures adaptées pour décrire les modèles dans le contexte de la IDM [78]. Ainsi, l'adoption des GNNs pour les problèmes de IDM est naturelle. De plus, des travaux récents montrent des utilisations intéressantes des GNNs pour différentes finalités en IDM [79, 80]. López et Cuadrado propose la génération de modèles structurellement réalistes via une architecture qui combine le codage de modèles réels en opérations d'édition et la génération subséquente de nouveaux modèles synthétiques similaires aux originaux. La combinaison d'un GNN et d'un Recurrent Neural Network (RNN) est au cœur de leur approche [79]. Les GNNs sont également au cœur des travaux de Di Rocco et al., qui les utilise pour établir un système de recommandation permettant la création efficace de modèles, réduisant ainsi les ajustements manuels dans des modèles complexes [80].

En plus des techniques basées sur les graphes, des modèles de langage pré-entraînés ont été utilisés dans le contexte de la IDM avec des résultats intéressants, comme dans les travaux de Weyssow et al. [81] pour assister la conception de métamodèles, ou dans le travail de Hernández López et al. [82], où les auteurs ont entraîné un modèle de langage avec un vocabulaire spécifique pour les activités de modélisation. Ces approches illustrent la diversité des cas d'utilisation des modèles de langage dans la IDM. Les Grand Modèle de Langages (GMLs) sont principalement appliqués dans la IDM pour fournir des capacités avancées de recommandation et de génération de modèles. Par exemple, des approches existantes visent à utiliser les GMLs pour proposer des recommandations de conception [83]. Dans l'ensemble, diverses analyses de la pertinence et de la performance des GMLs pour soutenir les activités de IDM montrent des résultats prometteurs [84, 85]. Cámara et al. montre une enquête sur les applications potentielles des GMLs pour aider dans différentes activités de modélisation, concluant qu'ils peuvent compléter le travail des

modélisateurs, notamment pour traiter des éléments concrets. Ils ont également présenté certaines limitations des GMLs lorsqu'il s'agit de concepts plus abstraits. Chen et al. a tenté de démontrer comment les GMLs peuvent automatiser entièrement la modélisation de domaine sans intervention humaine. Ils concluent que, de manière générale, une automatisation complète est peu pratique. Cependant, avec les capacités impressionnantes des GMLs, il est utile d'explorer d'autres stratégies pour améliorer leur utilisation, comme le Ingénierie de Prompt (IP), par exemple.

En résumé, la communauté IDM utilise de plus en plus les techniques de AA pour traiter différents problèmes de modélisation, avec un fort intérêt pour les représentations en graphe de modèles et les modèles de langage. Malgré cet intérêt croissant pour le AA dans les activités de IDM, il manque encore une solution spécifique pour la création et la maintenance des vues de modèles. La section 2.2 détaille les concepts importants autour du AP, et la Section 3.2 traite spécifiquement de son utilisation actuelle dans la IDM, en se concentrant sur les défis ouverts pour l'application des GMLs et des GNNs aux vues de modèles.

Le projet AIDOaRt

Ce travail de doctorat s'inscrit dans le cadre du projet **AIDOaRt** (**AI**-augmented **A**utomation for **D**ev**O**ps : **a** model-based framework for continuous development at **R**un**t**ime in cyber-physical systems) [29].

L'objectif global d'AIDOaRt est de soutenir efficacement l'ingénierie des systèmes tout au long de leur cycle de vie, depuis les exigences initiales jusqu'aux phases de test et de déploiement, en mettant l'accent sur le développement des SCPs. L'architecture conceptuelle d'AIDOaRt, illustrée dans la figure i, propose d'intégrer et de traiter divers types de données—telles que les données d'exécution (par exemple journaux de surveillance) et les données de conception (par exemple modèles, documentation, code)—afin de créer une représentation unifiée basée sur des modèles et stockée dans un référentiel partagé. Le projet propose d'utiliser les techniques de IDM pour fournir un cadre basé sur des modèles, construit avec des outils permettant de collecter et d'analyser les données d'exécution et de conception, en vue de solutions dédiées augmentées par l'Artificial Intelligence (AI). Les outils et solutions basés sur l'AI visent à soutenir les pratiques DevOps, combinant efficacement les opérations logicielles et informatiques pour le développement des SCPs. En résumé, la boîte à outils augmentée par l'AI surveille, analyse et automatise les tâches de développement et opérationnelles dans un contexte AIOps. Elle aborde des préoccu-

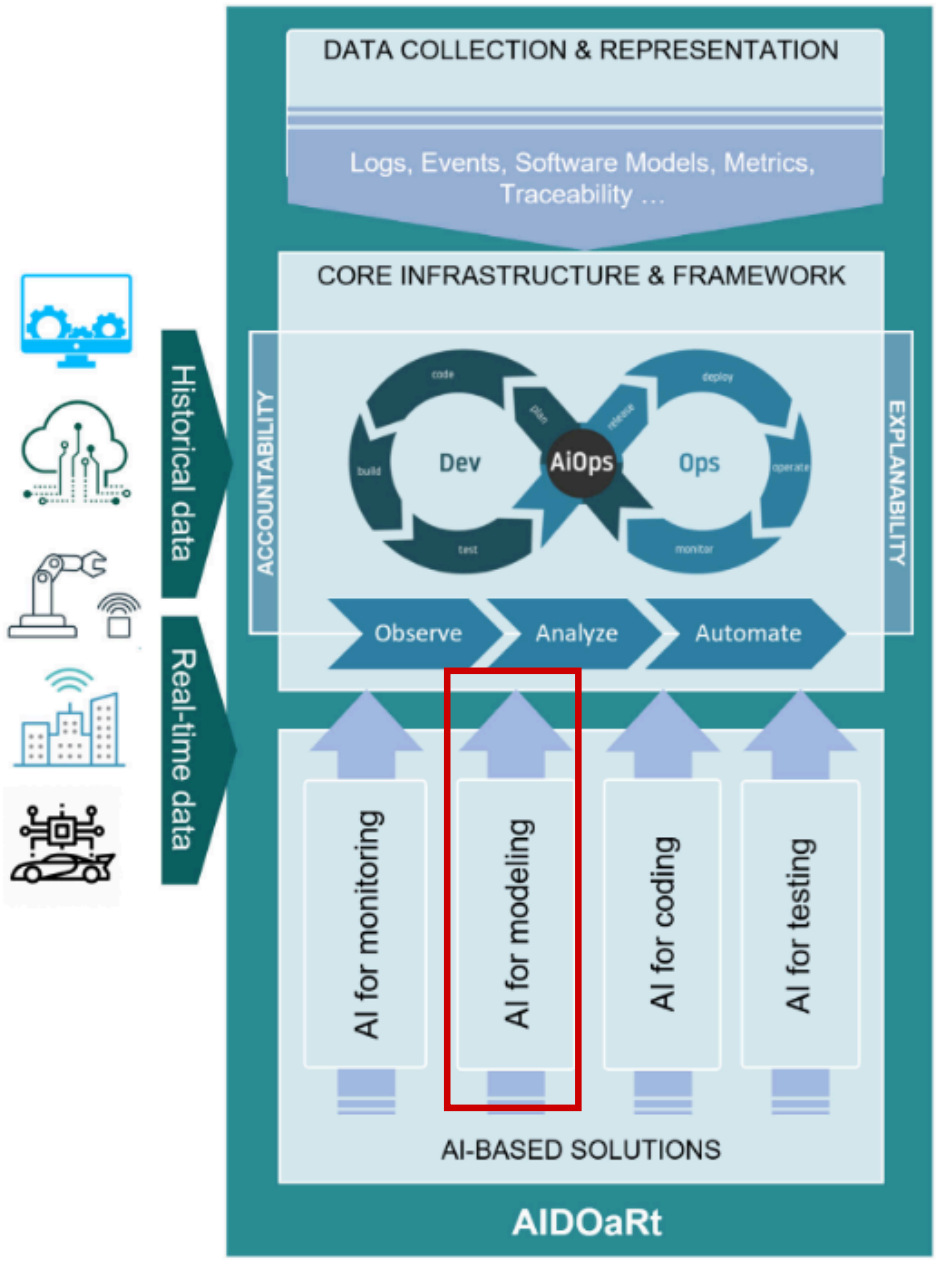

FIGURE 1 – Vue d'ensemble de l'architecture conceptuelle d'AIDOaRt. Le rectangle rouge met en évidence la position de cette thèse (Figure adaptée de [29, p. 5]).

pations transversales comme la responsabilité et l'explicabilité, en présentant aux parties prenantes des modèles d'exécution et de conception conformément aux principes de IDM.

Dans le contexte du projet, les partenaires *fournisseurs de solutions* développent les outils mentionnés ci-dessus, qui sont ensuite testés par les *fournisseurs de cas d'utilisation* à travers la création de démonstrateurs. Ces cas d'utilisation impliquent l'intégration des outils dans les processus d'ingénierie et de développement, la création des démonstrateurs, et l'évaluation des résultats du projet en mesurant l'efficacité des outils et des démonstrateurs. Pour le contexte d'AIDOaRt, la présente thèse fait partie des résultats du fournisseur de solutions IMT Atlantique (IMTA). La figure i illustre la position de notre travail dans le cadre général du projet, avec un rectangle rouge mettant en évidence les aspects « AI pour la modélisation » du projet.

Parmi les différents défis inhérents à l'ingénierie des SCPs modernes, caractérisés par leur complexité, leur interconnexion et leur dépendance croissante aux logiciels [10, 11], le projet AIDOaRt porte un intérêt particulier à certains défis spécifiques. Faciliter le développement continu, extraire de la valeur des données du système et permettre une collaboration efficace sont au cœur du projet [29] et sont particulièrement intéressants pour cette thèse. L'avènement de DevOps a introduit le besoin d'une continuité fluide entre la conception des systèmes et leur exécution, nécessitant une intégration continue et des boucles de rétroaction entre les phases de développement et d'exploitation. Les SCPs génèrent une quantité énorme de données à la fois en exécution et en conception. L'extraction d'informations significatives à partir de ces données est cruciale pour évaluer la validité des systèmes, prévoir les problèmes potentiels et prendre des décisions d'ingénierie éclairées. Le développement de SCPs complexes implique souvent des équipes distribuées à travers diverses disciplines d'ingénierie. Cela nécessite des mécanismes de collaboration efficaces, une compréhension partagée des modèles de système, et des canaux de communication rationalisés pour garantir un processus d'ingénierie fluide et productif.

Comme présenté précédemment, le IDM et l'AI ont tous deux démontré leur efficacité dans des applications industrielles complexes [28, 33, 34]. Le projet AIDOaRt considère le IDM et l'AI comme des forces complémentaires et synergiques pour faire progresser l'ingénierie des SCPs complexes. Le projet exploite les forces des deux domaines pour relever les défis liés à la gestion de la complexité des systèmes, à la facilitation du développement continu, à l'extraction de valeur des données des systèmes, et à la garantie d'une collaboration efficace. Cette thèse est alignée sur les principaux objectifs du projet, visant à tirer parti des applications de l'AI, notamment des GMLs et des GNNs, pour

améliorer une solution de vues de modèle.

Énoncé du Problème

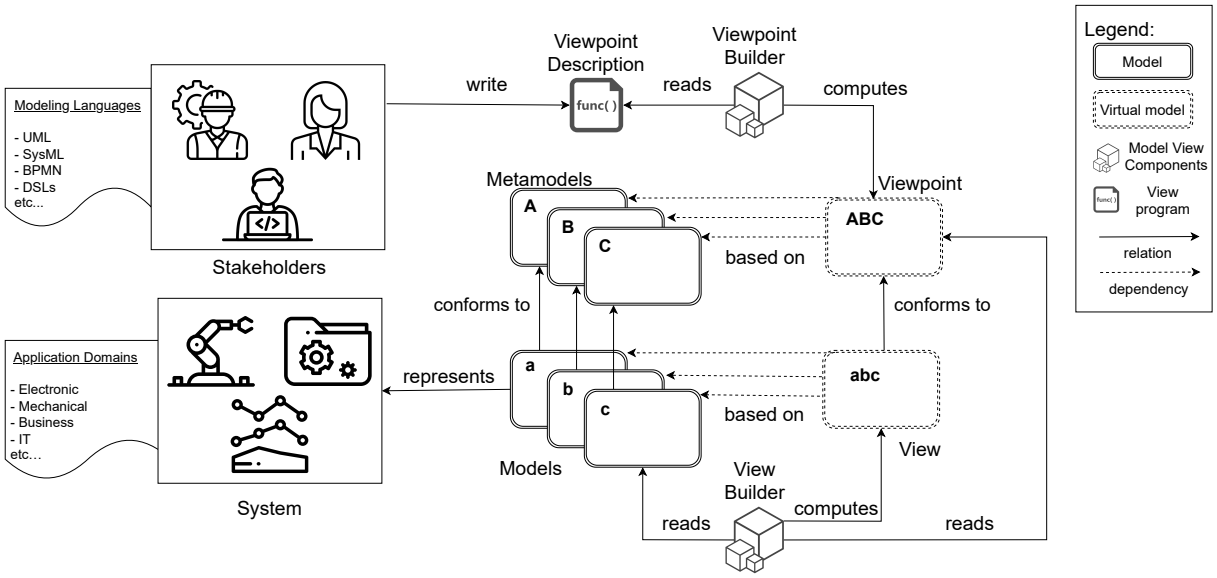

FIGURE II – Aperçu de la terminologie des Model Views et de son application dans un scénario IDM

La complexité croissante de l'ingénierie des SCP a conduit au développement de nombreux langages de modélisation et outils spécialisés, chacun adapté à un aspect spécifique du système [10, 12]. Les modèles développés, créés pour représenter divers sous-systèmes ou perspectives (par exemple, logiciels, matériel, interactions utilisateurs, analyses de sécurité), proviennent souvent de domaines variés et reposent sur des méta-modèles et paradigmes différents [21]. En conséquence, les ingénieurs sont confrontés à des défis majeurs pour intégrer ces modèles hétérogènes en une représentation cohérente du système global [19, 21, 59].

Parmi les différentes approches proposées pour relever ces défis, les *model views* se sont imposées comme une solution prometteuse, permettant aux ingénieurs d'extraire et de présenter des sous-ensembles spécifiques d'informations provenant de plusieurs modèles d'une manière pertinente à leurs préoccupations particulières [64]. Cependant, le développement de model views reste une tâche complexe. En général, les ingénieurs doivent définir manuellement les relations entre les modèles. Ces définitions nécessitent souvent une expertise approfondie du domaine ainsi qu'une maîtrise des transformations et des requêtes

de modèles, rendant difficile leur automatisation ou leur généralisation à travers différents projets. Cette complexité constitue un frein majeur à l'adoption généralisée des model views dans la pratique, soulignant le besoin de meilleures méthodes et outils pour soutenir leur génération et leur maintenance automatiques. En résumé, le problème que nous visons à traiter dans cette thèse est de mieux accompagner les ingénieurs de vues [9] en les aidant à définir les vues et à automatiser leur construction autant que possible.

La figure ii illustre la terminologie principale des model views tout en résumant les problèmes mentionnés ci-dessus. Différents professionnels (i. e. parties prenantes) de divers domaines utilisent des outils de modélisation pour créer des représentations virtuelles de systèmes complexes. Un système donné est décrit par divers modèles qui se conforment potentiellement à différents méta-modèles. Ces modèles peuvent être créés en utilisant différents langages de modélisation en fonction du domaine du système modélisé. Des exemples incluent l'Unified Modeling Language (UML) pour la création de modèles logiciels [86], le Business Process Model and Notation (BPMN) pour modéliser les processus métier [87], et le Systems Modeling Language (SysML™) [88] pour les modèles de systèmes-de-systèmes.

La *Description de Vuepoint* est généralement un programme écrit avec un LD dédié utilisé pour décrire comment combiner ces modèles en un artefact de modélisation unique.

Au niveau des méta-modèles, un *Vuepoint* détermine quels concepts et propriétés des méta-modèles contributifs doivent être inclus ou exclus dans les vues correspondantes. Il exprime également comment ces concepts doivent être interconnectés, i. e. avec quelles règles de combinaison. Au niveau des modèles, une *Vue* combine un ensemble donné de modèles contributifs selon ce vuepoint. Il convient de noter que, dans cette thèse, les vuepoints et les vues fonctionnent respectivement comme des méta-modèles et modèles virtuels. Un modèle (respectivement, méta-modèle) virtuel ne fait que pointer vers des éléments des modèles (respectivement, méta-modèles) originaux, évitant ainsi toute duplication d'information inutile. Dans ce contexte, un système de model views se divise en deux composantes principales : le *Générateur de Vuepoint* et le *Générateur de Vue* qui, respectivement, calculeront (i. e. compileront) le vuepoint et la vue.

Gérer l'hétérogénéité des modèles sous-jacents lors de la création de ces vues est un défi bien connu [42]. Cependant, cela repose souvent sur les connaissances des experts du domaine pour définir à la fois les propriétés à sélectionner et les règles de combinaison

9. La partie prenante responsable de la création et de la fourniture de la vue aux autres parties prenantes (internes ou externes). Ce sont les acteurs qui exécutent le développement des vues de modèle.

qui lient les modèles lors du calcul de la vue. Nous visons à utiliser les techniques de AP pour faciliter le processus d'écriture de la description du vuepoint et son calcul, aidant ainsi les ingénieurs de vues tout au long du processus de développement des model views.

Dans ce contexte, cette thèse vise à valider les bénéfices pratiques de l'approche proposée en répondant aux questions de recherche suivantes :

— **QR1** : Au niveau des méta-modèles, comment les techniques de AP peuvent-elles être appliquées pour automatiser ou assister la définition des model views ?
— **QR2** : Au niveau des modèles, comment les techniques de AP peuvent-elles être appliquées pour aider à calculer automatiquement les model views ?
— **QR3** : Quels sont les bénéfices pratiques et les limites des méthodes basées sur le AP pour soutenir la définition et l'automatisation des model views ?

La réponse à ces questions de recherche vise à explorer et identifier la faisabilité de l'application des mécanismes de AP pour améliorer les fonctionnalités des model views au sein des outils IDM, à évaluer empiriquement les améliorations ou limitations introduites par leur utilisation, et à vérifier comment cela impacte leur utilisation dans un contexte industriel, grâce au cadre AIDOaRt. En outre, il s'agira de montrer comment permettre la participation efficace des spécialistes en AA au développement des model views, en intégrant leur expertise de manière fluide dans les processus de modélisation.

Méthodologie et Contributions

Dans cette thèse, nous contribuons à répondre aux questions de recherche définies (cf. section 1.2) par la conception et l'implémentation d'une ***solution améliorée pour les model views, alimentée par des techniques d'apprentissage profond***. Pour mener cette recherche, nous avons suivi les principes de la Design Science Research (DSR) [89]. Cette méthodologie est appropriée pour notre travail, car elle met l'accent sur (i) le développement et l'amélioration d'artefacts informatiques (Information Technology (IT)) utiles, et (ii) une validation empirique via une mise en œuvre pratique.

Pour le point (i), notre travail propose des versions améliorées d'outils de modélisation. Plus spécifiquement, le travail propose d'utiliser à la fois des GMLs et des GNNs appliqués à l'outil EMF Views.

Pour le point (ii), nous fournissons une validation empirique via la mise en œuvre pratique de la solution proposée dans le contexte des cas d'usage fournis par un partenaire industriel collaborant au projet AIDOaRt.

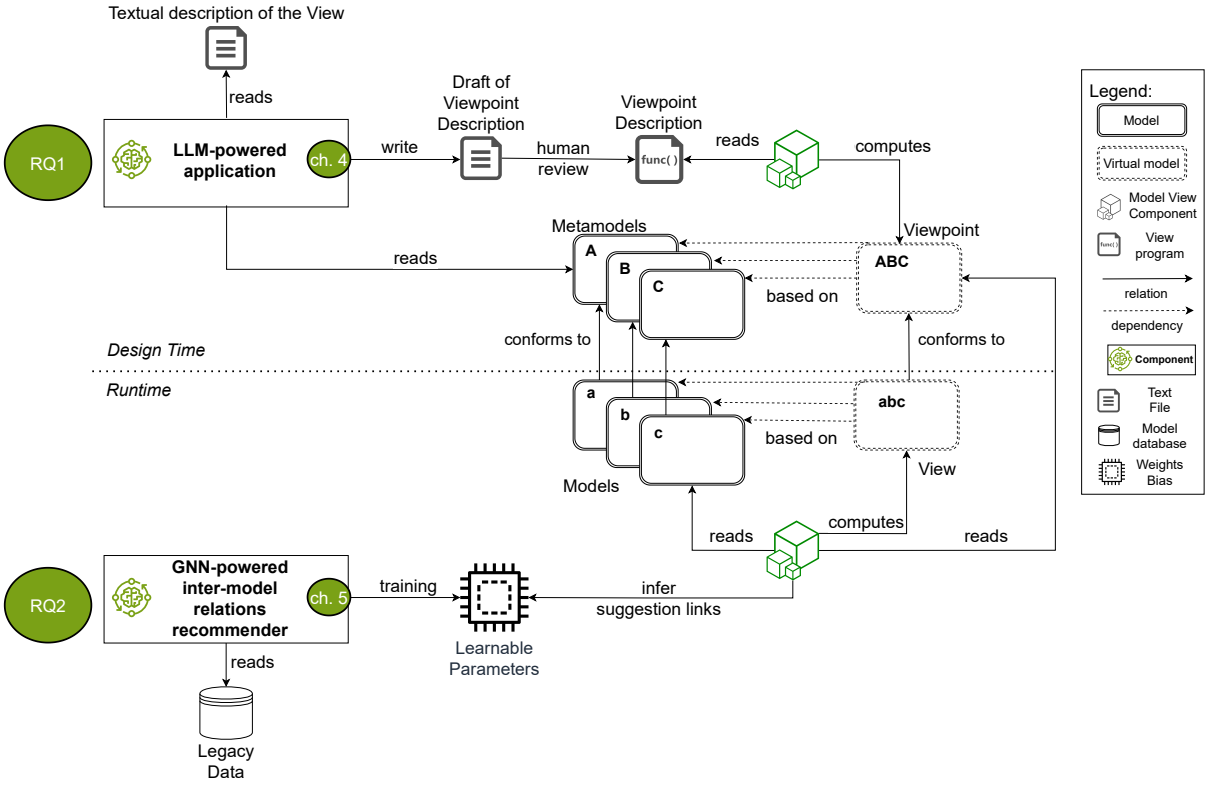

FIGURE III – Aperçu des contributions de la thèse dans un cadre intégré

En résumé, les six activités principales de la méthodologie DSR sont :

1. *Identification et motivation du problème* : Nous avons étudié les travaux en IDM, avec un focus sur la fédération de modèles et plus particulièrement sur les model views, en recherchant l'application de techniques d'AI. Cela nous a permis d'avoir une vue d'ensemble du domaine, de concevoir notre motivation et d'identifier les forces et limitations. De plus, nous avons identifié un cas d'usage adapté parmi tous les partenaires du projet AIDOaRt pour l'utilisation des model views.
2. *Définition des objectifs pour une solution* : Nous avons identifié les besoins pour fournir des outils de modélisation enrichis par l'AP pour la création de model views.
3. *Conception et développement de l'artefact* : Nous avons détaillé l'approche sur la manière d'utiliser les GMLs dans la phase de conception des model views ainsi que l'utilisation des GNNs pour calculer efficacement ces vues lors de leur exécution.
4. *Démonstration par des cas d'usage* : Nous avons implémenté les deux approches pour l'outil ***EMF Views***, en les illustrant avec des exemples issus de la littérature.
5. *Évaluation selon des critères prédéfinis* : Nous avons évalué notre solution de manière empirique en adaptant la stratégie de *Technical Risk & Efficacy* proposée par Venable et al. [90]. Pour chaque contribution, une évaluation formative a été réalisée en utilisant des exemples et des métriques adaptés.
6. *Communication des résultats* : Les résultats de cette thèse ont été publiés comme décrit dans la section 1.5.

En suivant la méthodologie établie, la principale contribution de cette thèse est d'améliorer les outils de modélisation existants en intégrant des techniques d'apprentissage profond. La figure iii fournit un aperçu conceptuel de l'approche proposée. Elle illustre deux composants clés basés sur l'AP (propulsés par des GMLs et des GNNs) qui collaborent pour construire et affiner des model views au sein d'un flux de travail en plusieurs phases, englobant à la fois les étapes de conception et d'exécution.

Plus précisément, les GMLs assistent les ingénieurs en vues durant la phase de conception en permettant la définition des vues avec un minimum d'efforts. De plus, les GNNs facilitent la participation des spécialistes en AA dans la définition des model views, en identifiant les liens intermodèles et en améliorant ainsi l'expressivité et l'utilité des vues.

La partie supérieure de la figure iii est un diagramme à haut niveau montrant comment les GMLs sont utilisés pendant la phase de conception pour rédiger une première description du point de vue. Ce brouillon est ensuite raffiné via une revue humaine et compilé en un point de vue final. L'interface en langage naturel des GMLs permet aux ingénieurs, quel que soit leur niveau d'expertise, de contribuer en fournissant une *des-*

cription textuelle de la vue sans connaissance approfondie des modèles sous-jacents. Cet apport constitue la première contribution majeure de la thèse : une *application basée sur les GMLs* pour faciliter la définition des descriptions des points de vue. Cette contribution aide à répondre à la **QR1**. Elle est détaillée au chapitre 4.

Une fois la phase de conception terminée, le défi se déplace vers l'exécution, où l'identification efficace des liens intermodèles est cruciale. Ce défi peut découler des limites des capacités de requêtage du LD utilisé ou d'une connaissance limitée des modèles par les ingénieurs. Lorsqu'une quantité suffisante de *données de modélisation héritées* est disponible, nous pouvons entraîner un Neural Network (NN) pour identifier et recommander des liens potentiels, même lorsqu'ils ne sont pas explicitement exprimés.

La partie inférieure de la figure iii met en avant le *recommander de relations intermodèles basé sur des GNNs*, qui applique des GNNs pour la prédiction de liens dans les model views. Les GNNs sont particulièrement adaptés à cette tâche grâce à leur capacité à gérer des données structurées en graphe, ce qui les rend idéaux pour les modèles en IDM. Cette contribution aide à répondre à la **QR2**. Elle est détaillée au chapitre 5.

Comme mentionné précédemment, nous avons implémenté ces deux contributions dans le plugin *EMF Views* [64]. Conformément à la méthodologie établie, le plugin EMF Views enrichi constitue un artefact informatique significatif développé pour relever les défis identifiés. Nous l'avons utilisé dans un cas d'usage industriel comme exemple motivant, démontrant ses applications et avantages pratiques (cf. 2.3). Lorsqu'appliquée à EMF Views, l'analyse globale de ces deux contributions nous aide à répondre à la **QR3**, car elle permet de vérifier ses avantages et ses limitations.

Table of Contents

Part I

Introduction and General Context

Chapter 1

Introduction and Context

1.1 General Context

Digital transformations drive the world. This is especially evident when we investigate the evolution in the development of the Internet of Things (IoT) [1] systems and the growth of very complex digital systems, such as Cyber-Physical-Systems (CPSs) [2]. Essentially, CPSs integrates the software components and related artifacts with the physical elements using a set of sensors and actuators that should constantly communicate with each other in ways that change with context. With the advent of *Industry 4.0*, modern CPSs have advanced not only to collect and manage data but also to support traceability mechanisms that help real-time monitoring and data lineage [2–4]. This traceability aims to facilitate the retrieval of relevant information across different stages of data processing and system operations [3, 4]. Examples of CPSs exist in different domains, e. g. industrial control systems [5], smart businees [6], health-care [7, 8], and avionic systems [9].

CPS engineering is often challenging since the ability to efficiently automate activities is primarily based on the quality and relevance of the available data and tools [10]. Developing CPSs from scratch is challenging, as engineers must handle several issues, including data heterogeneity, multi-stakeholder environment, and concurrency or time-based constraints, for example [11]. A key challenge of this kind of system that we want to highlight is the fragmentation of information between the various needs of stakeholders and their different profiles and levels of expertise [11, 12]. Proper tools are needed to support data traceability and federation to improve CPS engineering processes and overcome this challenge.

Model-Driven Engineering (MDE) is an engineering approach in which models are used as the primary artifact, which can help to foster knowledge and data reusing in process automation [13]. It allows different representations of the same system (or part of the system) through convenient abstractions with techniques and tooling to manipulate these abstractions [13–15]. MDE provides suitable ways of tackling the challenges of

engineering CPSs mentioned before [10, 15–18]. However, some limitations remain, e. g. proper abstractions to model physical components into computer abstractions [10], designing domain-specific simulation tools [15], and providing tool support for automated federation [19] to cite a few examples. The present thesis highlights the challenge of federating multiple models or viewpoints into a cohesive whole, which requires sophisticated tool support [19, 20]. Automated federation helps integrate disparate models, ensuring consistency and interoperability, which are crucial for large-scale and multi-disciplinary projects but is still an area with limited support [19, 21].

The Artificial Intelligence (AI) brings a set of algorithms and methods for automating solutions to complex problems through predictions, helpful data-driven information, and recommendations [22–24]. The latest advancements in AI impact all aspects of system development, from specification to maintenance, affecting their design, validation, and deployment. Among all the sub-fields of AI, this work is interested in Machine Learning (ML) and, more precisely, in Deep Learning (DL) techniques [25], which use Artificial Neural Networks (ANNs) to learn complex patterns and predict unseen scenarios. This particular interest is given by the recent fact that among all techniques studied under the AI cover, the recent breakthroughs are very commonly backed by some variation of DL, even when combined with some other ML technique [26, 27] This AI expansion also contributes to the development of CPSs, helping to solve problems related to decision-making and enabling more precise predictions because AI can analyze the volume of data generated in low-latency [24]. Given this scenario, DL can be used to enhance traditional MDE techniques and thus improve the current support for CPS engineering providing the necessary tool support for remaining challenges.

In recent years, in response to the aforementioned challenges, different industrial-academia partnerships were put in place to enhance the quality of development, integration, and maintenance of CPSs engineering, e. g. the MegaM@Rt2 project [28] [1] and the AIDOaRt project [29]. [2] The first was about using MDE to handle the integration of design-time and runtime aspects of a given system, demonstrating the capabilities of using MDE in the CPS engineering process and also identifying some of the limitations mentioned above [30–32]. The AIDOaRt project has given rise to this thesis's main contributions. It proposes developing a model-based framework to analyze runtime and design-time data to dedicated AI-augmented solutions. Aligned with our objectives, we

1. `https://megamart2-ecsel.eu/` *(Last Accessed in November 2024)*
2. `https://www.aidoart.eu/` *(Last Accessed in November 2024)*

can get examples of other successful projects handling industrial projects using both MDE (e. g. iDev40 [33] [3]) and AI(e. g. AI4DI [34] [4]), so it is possible to see that there is room to combine both, aiming to solve different challenges for CPS engineering.

This thesis aims to show a possible path to combine MDE and DL, addressing one pain point of the CPS development: how to assist engineers when combining different models (i. e. different views of the same system) during various activities of the CPS engineering process. We want to help modelers effectively combine models into *model views*, allowing a range of stakeholders to participate effectively in the system's development and use. More precisely, we want to show the feasibility of profiting from the power of AI to enhance modeling activities, mainly the model views creation. This introductory chapter provides the necessary high-level background on MDE and AI (namely DL as a subfield of ML). Section 1.2 details the problem we propose to help solving. Section 1.3 shows how we propose to address the problem. It presents the methodology and a high-level view of our two main contributions. Finally, section 1.4 and 1.5 presents the general organization of the thesis and the scientific output achieved during its realization.

1.1.1 Model-Driven Engineering and Model Views

Creating models to help understanding different systems is at the core of many scientific domains. The idea is to use abstractions to represent a system (or part of it) so that the representation (i. e. the model) answers questions in place of the actual system [35]. In this thesis, we are highly interested in modeling as an essential part of systems and software engineering, mainly on complex and multi-stakeholder systems like the case of CPSs.

MDE is an approach that intends to apply modeling methodologies and tooling to systems engineering, using models as first-citizen artifacts [13]. In the MDE context, a *system* is a real-world artifact (e. g. a complex system as a CPS) that models can represent. The *models* are abstract representations of a system used to plan, analyze, investigate, and sometimes generate (parts of) the system. To create these models, they should follow the structure and rules established by a given *metamodel*. These metamodels define specific *modeling languages* that allow users to create models according to well-defined syntax and semantics. These languages are often referred to as Domain-Specific Languages (DSLs) because they are tailored to particular application domains or types of systems. It is

3. `https://www.idev40.eu/` *(Last Accessed in November 2024)*
4. `https://ai4di.eu/` *(Last Accessed in November 2024)*

possible to convert one model into another, following a set of rules expressed in a *model transformation*. Transformations are helpful to refine, translate, report, or analyze the modeled system.

Altough simpler systems can be represented by simple models, complex systems often require more numerous and complex models, commonly written with different modeling languages (i. e. metamodels). Each modeling language has emerged to address specific needs across different domains. This proliferation has increased the heterogeneity within the modeling ecosystem, creating challenges for interoperability and integration [36], which led to situations that demand different *(meta)model management* strategies [20, 37]. A key management challenge arises when different stakeholders with various levels of expertise and a wide range of requirements create, manage, and use models written in multiple languages for specific purposes [21]. Different strategies were proposed to handle this *model heterogeneity*, e. g. using model transformations to bring all metamodels under one operating paradigm [38], the composition of (meta)models [39] and unification using intermediate languages [40].

One possible way to cope with this challenge is to use what is called *model federation* [36, 41], which means that models developed by different teams or stakeholders should be linked to each other providing a mechanism for the cohesive integration of heterogeneous models [36, 42].

Several strategies achieve model federation in MDE. These techniques usually aim to facilitate integrating, synchronizing, and managing models from different domains or tools while maintaining their independence. We have model merging, megamodeling, collaborative modeling, and model weaving, to cite a few examples. Model merging combines two or more models into one model, resolving conflicts and inconsistencies [43, 44]. Megamodeling manages collections of models, defining semantic relationships between them and keeping a high-level description of how models interact without altering their internal structures [28, 45–47]. Collaborative modeling supports model federation by enabling multiple stakeholders to collaborate on models using shared representations and mechanisms to synchronize changes across distributed environments [48–50, 91, 92]. Model weaving refers to integrating multiple models or metamodels into a unified representation by establishing explicit links between their elements, enabling modularity and reusability of models [51–54]. These techniques can be combined between them to achieve an efficient model federation [53, 55] and also combined with other non-MDE techniques, like the combination with DevOps [54], for example. It is also worth mentioning that model

federation is the target of an industrial standard through the "ISO/TC 184/SC 5" [56].

Finally, we have *Model views* as another strategy to cope with model federation. Essentially, it addresses the challenge of information fragmentation by presenting only the relevant information to each stakeholder [21]. A model view is a single modeling artifact (i. e. a representation of a system) compound with elements coming from different models. Eventually, they are completed with interconnections between them and additional data, either manually entered or automatically computed [42, 57]. In other words, a model view represents the system from a specific perspective given a viewppoint [21, 57, 58].

On that matter, different proposals appeared in the modeling community to implement the concept of the model federation through model views [19, 21], effectively implementing the ideas of the *multi-view software and system modeling* paradigm [19, 59, 60]. For the scope of this thesis, we are especially interested in multi-view modeling through solutions based on model views. We narrow our analysis to solutions that potentially include (i) dedicated language(s) for view(point) description and/or (ii) virtualization mechanisms for inter-model relations. To illustrate with a few examples, we can cite OpenFlexo [61], [5] VIATRA [62, 63], [6] and EMF Views [64]. [7] Details of these solutions and other complementary ones can be found in section 3.1.

Model views technology and application are at the core of this thesis, and we use *EMF Views* as the tool of choice to study and prototype our proposed contributions. EMF Views has been developed in our research group (Naomod [8]) in the last years [42, 64, 65], and it is the tool used in the context of AIDOaRt project (cf. section 1.1.3), which underlines its practical relevance for CPSs' engineering. EMF Views have already shown its feasibility in different industrial applications like the integration of safety-critical components for software systems [65] and the engineering of construction equipament [66], for example. We provide all the necessary tool details in section 2.1.3. EMF Views were built with explicit expressiveness, inspired by database views and non-intrusive mechanisms that do not demand changes in the base models, effectively applying virtualization mechanisms. A comparative analysis between EMF Views and other model view solutions features are provided in section 3.1.

5. `https://openflexo.org/` *(Last Accessed in November 2024)*
6. `https://eclipse.dev/viatra/` *(Last Accessed in November 2024)*
7. `https://www.atlanmod.org/emfviews/` *(Last Accessed in November 2024)*
8. `https://naomod.github.io/` *(Last Accessed in November 2024)*

1.1.2 Artificial Intelligence and Model-Driven Engineering

In the context of this thesis, the main goal of AI is to help engineers produce systems faster and with improved quality, handling ever more complex problems [67]. To do so, it mainly uses DL techniques [25]. This is true for many applications, including vision, speech recognition and generation, natural language processing, image and video generation, multi-agent systems, planning, decision-making, and integration of vision and motor control for robotics [67]. It also has been the *de facto* technique used for software engineering [68].

On the MDE context, a non-exhaustive list of ML applications include model repair [69], automatic classification of metamodel repositories [70], solving conflict in model merging [93], and automatic requirements extraction to use in MDE context [71]. Indeed, the correct use of ML is an identified challenge for the next steps in MDE evolution [20].

Barriga *et al.* [72] found a wide range of ML techniques to deal specific with model-repair activities, from rule-based ML [73] to decision trees [74] and also the use of DL through ANNs [75]. According to the authors, while rule-based methods provide solutions tailored to specific constraints in model repair (e. g. resolving cardinality violations) [72, 73], decision trees balance simplicity and effectiveness, making them suitable for many automated repair scenarios [72, 74, 76]. Particularly within DL, the use of ANNs allows handling complex dependencies in model elements, enhancing the adaptability and scalability of solutions for model repair.

A latent ANN architecture for modeling problems is the use of Graph Neural Networks (GNNs) [77]. It aligns well with MDE requirements since labeled graphs are suitable structures to describe models in the MDE context [78]. Given that, adopting GNNs for MDE problems is straightforward. Moreover, recent research efforts show interesting uses of GNNs for different purposes in MDE [79, 80]. López and Cuadrado proposes the generation of structurally realistic models through an architecture that combines the encoding of real models into edit operations and the subsequent generation of synthetic new models similar to the original ones. The combination of a GNN and Recurrent Neural Network (RNN) is at the core of their approach [79]. GNNs are also at the core of the work of Di Rocco *et al.*, which uses them to establish a recommendation system to allow the efficient creation of models, reducing manual adjustments in complex models. [80].

Besides graph-based techniques, pre-trained language models have been used in the MDE context with some interesting results like in the work of Weyssow *et al.* [81] to assist design of metamodels or in Hernández López *et al.* [82] where authors trained a language

model with specific vocabulary for modeling activities. These approaches exemplify the diversity of use cases for language models in MDE. Large Language Models (LLMs) are primarily applied within MDE to provide advanced model recommendation and generation capabilities. For example, existing approaches intend to use LLMs to propose design recommendations [83]. Overall, various analyses of the relevance and performance of LLMs for supporting MDE activities show promising results [84, 85]. Cámara *et al.* shows an investigation of the potential applications of LLMs in helping with different modeling activities, concluding that they can complement the modeler's work mainly when dealing with concrete elements. They also presented some limitations of LLMs when dealing with more abstract concepts. Chen *et al.* tried to show how LLMs can fully automate domain modeling without human interaction. They conclude that, in general, full automation is impractical. Still, with the impressive capabilities of LLMs, it is worth investigating other strategies to improve their use, like Prompt Engineering (PE), for example.

In summary, the MDE community increasingly uses ML techniques to cope with different modeling problems, with a high appeal for graph representations of models and language models. Despite this increasing interest in ML for MDE activities, it still lacks a solution specifically for creating and maintaining model views. Section 2.2 details important concepts around DL, and section 3.2 specifically addresses its current use on MDE, focusing on the open challenges for applying LLMs and GNNs for model views.

1.1.3 The AIDOaRt Project

This PhD work has been realized in the context of **AIDOaRt**(**AI**-augmented **A**utomation for **D**ev**O**ps: **a** model-based framework for continuous development at **R**un**t**ime in cyber-physical systems) [29].

The overall idea of AIDOaRt is to efficiently support the system's engineering throughout its lifetime, from the initial requirements to testing and deployment, focusing on CPSs's development. The AIDOaRt conceptual architecture illustrated in the figure 1.1 proposes to integrate and process various data types—such as runtime data (e. g. monitoring logs) and design data (e. g. models, documentation, code) to create a unified model-based representation stored in a shared repository. The project proposes using MDE techniques to provide a model-based framework built with tooling to gather and analyze runtime and design-time data to dedicated AI-augmented solutions. The provided tooling and AI solutions intend to support DevOps practices, efficiently combining software and Information Technology (IT) operations for CPSs' development. In summary, the AI-

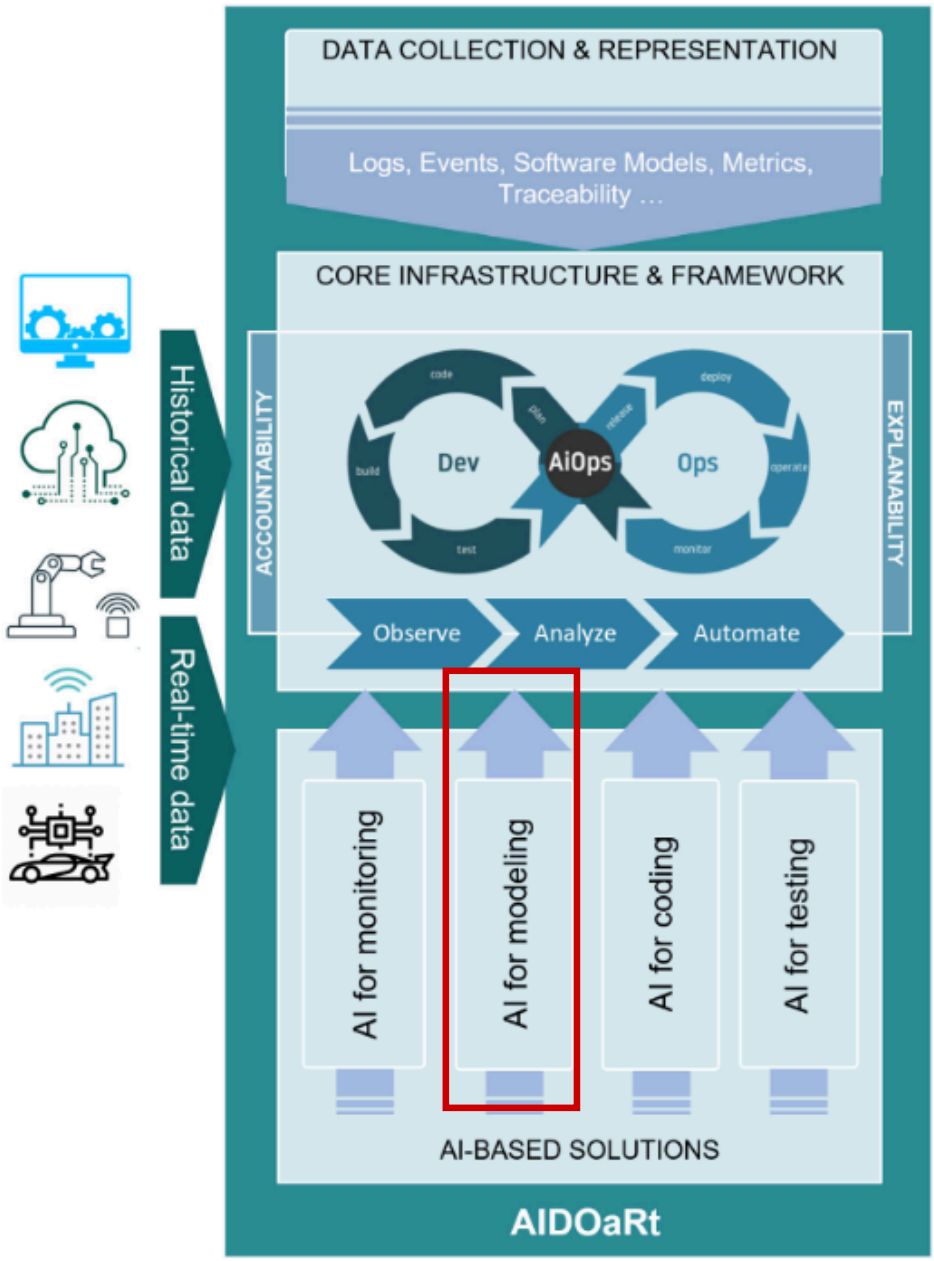

Figure 1.1 – An overview of AIDOaRt conceptual architecture. The red rectangle highlights where this thesis is positioned (Figure adapted from [29, p. 5]).

augmented toolkit monitors, analyzes, and automates the development and operational tasks in an AIOps context. It addresses cross-cutting concerns like accountability and explainability, presenting stakeholders with runtime and design models per MDE principles.

In the project context, the *solutions provider* partners develop the tooling as mentioned above, which is then tested by the *use case providers* through the creation of demonstrators. These use cases involve integrating the tools into engineering and development processes, creating the demonstrators, and evaluating the project's outcomes by assessing the effectiveness of both the tools and the demonstrators. For the AIDOaRt context, the present thesis is part of the outcome of the solution provider IMT Atlantique (IMTA). Figure 1.1 illustrates where our work is positioned in the overall project context with a red rectangle highlighting the "AI for Modeling" aspects of the project.

Among the different challenges inherent in engineering modern CPS, which are characterized by their complexity, interconnectedness, and increasing reliance on software [10, 11], the AIDOaRt project has a particular interest in some specific challenges. Facilitating continuous development, extracting value from system data, and enabling efficient collaboration are at the core of the project [29] and are especially interesting for this thesis. The advent of DevOps introduced a need for a seamless continuum between system design and runtime, which requires continuous integration and feedback loops between development and operational phases. CPSs generate enormous amounts of data at both runtime and design time. Extracting meaningful insights from this data is crucial for assessing system validity, predicting potential issues, and making informed engineering decisions. Developing complex CPSs often involves distributed teams across various engineering disciplines. This necessitates efficient collaboration mechanisms, a shared understanding of system models, and streamlined communication channels to ensure a smooth and productive engineering process.

As presented before, both MDE and AI have successful stories of its use along complex industrial applications [28, 33, 34]. The AIDOaRt project views MDE and AI as complementary and synergistic forces in advancing the engineering of complex CPSs. The project leverages the strengths of both domains to address the challenges of handling system complexity, facilitating continuous development, extracting value from system data, and ensuring efficient collaboration. This thesis is aligned with the main objectives of the project, aiming to leverage AI applications, namely LLMs and GNNs, to enhance a model view solution.

1.2 Problem Statement

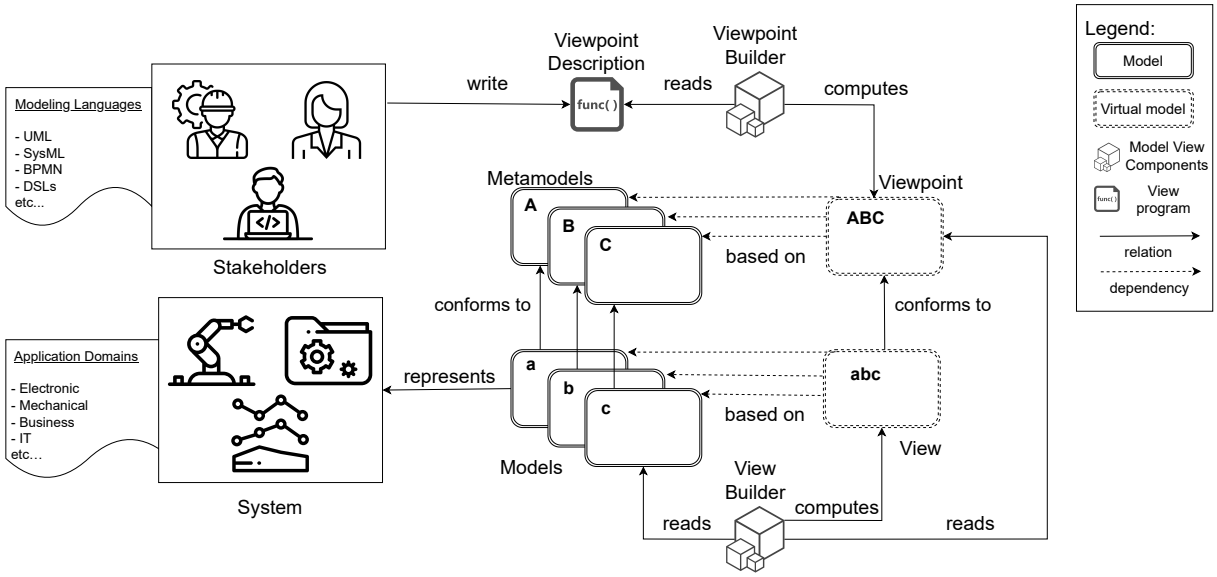

Figure 1.2 – An overview of the Model Views terminology and its application in a MDE scenario

The increasing complexity of the CPS engineering has led to the development of numerous specialized modeling languages and tools, each tailored to a specific aspect of the system [10, 12]. The developed models, created to represent various subsystems or perspectives (e. g. software, hardware, user interactions, safety analysis), often come from diverse domains and are based on different metamodels and paradigms [21]. As a result, engineers face significant challenges in integrating these heterogeneous models into a cohesive representation of the overall system [19, 21, 59].

Among the various approaches proposed to address these challenges, *model views* have emerged as a promising solution, allowing engineers to extract and present specific subsets of information from multiple models in a way relevant to their particular concerns [64]. However, the development of model views is a non-trivial task. Usually, engineers must manually define the relationships between models. These definitions often require deep domain knowledge and expertise in model transformation and querying, making it difficult to automate or generalize across different projects. This complexity presents a significant barrier to the widespread adoption of model views in practice, underscoring the need for better methods and tools to support their automatic generation and maintenance. In summary, the problem we aim to target in this thesis is how to support view engineers

better, helping them define views and automate the construction of those views as much as possible.

Figure 1.2 depicts the main terminology around model views while summarizing the abovementioned problems. Different professionals (i. e. stakeholders) in various domains use modeling tools to create virtual representations of complex systems. A given system is described by various models that potentially conform to different metamodels. These models can be created using different modeling languages depending on the domain of the modeled system. Examples include the Unified Modeling Language (UML) to create software models [86], the Business Process Model and Notation (BPMN) used to model business processes [87], and Systems Modeling Language (SysML™) [88] for systems-of-systems models. The *Viewpoint Description* usually is a program written with a dedicated DSL used to describe how to combine these models in a single modeling artifact.

At the metamodel level, a *Viewpoint* determines which concepts and properties from the contributing metamodels should be included or excluded in the corresponding views. It also expresses how these concepts should be interconnected, i. e. with which combination rules. At the model level, a *View* combines a given set of contributing models according to this viewpoint. It is worth mentioning that in this thesis, both viewpoints and views work as virtual metamodels and models, respectively. A virtual model (respectively, metamodel) only points to elements from the original models (respectively, metamodels), thus preventing unnecessary information duplication. In this context, a model views system splits into two main components: The *Viewpoint Builder* and the *View Builder* that will respectively compute (i. e. compile) the viewpoint and the view.

Dealing with the heterogeneity of the underlying models when creating these views is a well-known challenge [42]. Still, it often relies on the knowledge of domain experts to define both the properties to be selected and the combination rules that link the models when computing the view. We aim to use DL to help the process of writing the viewpoint description and, in its computation, helping view engineers during the whole process of model views development.

Given this scenario, this thesis aims to validate the proposed approach's practical benefits by answering the following research questions:

— **RQ1**: At the metamodel level, how can DL techniques be applied to automate or assist in defining model views?

— **RQ2**: At the model level, how can DL techniques be applied to help automatically compute model views?

— **RQ3**: What are the practical benefits and limitations of using DL-based methods for supporting the definition and automation of model views?

The response to these research questions intends to explore and identify the feasibility of applying DL mechanisms to improve model views functionalities within MDE tooling, empirically assess the performance improvement or limitations introduced by its uses, and check how it affects their use in an industrial context, thanks to the AIDOaRt project. Additionally, it should show how to enable the effective participation of ML specialists in model views development, integrating their expertise seamlessly into the modeling processes.

1.3 Methodology and Contributions

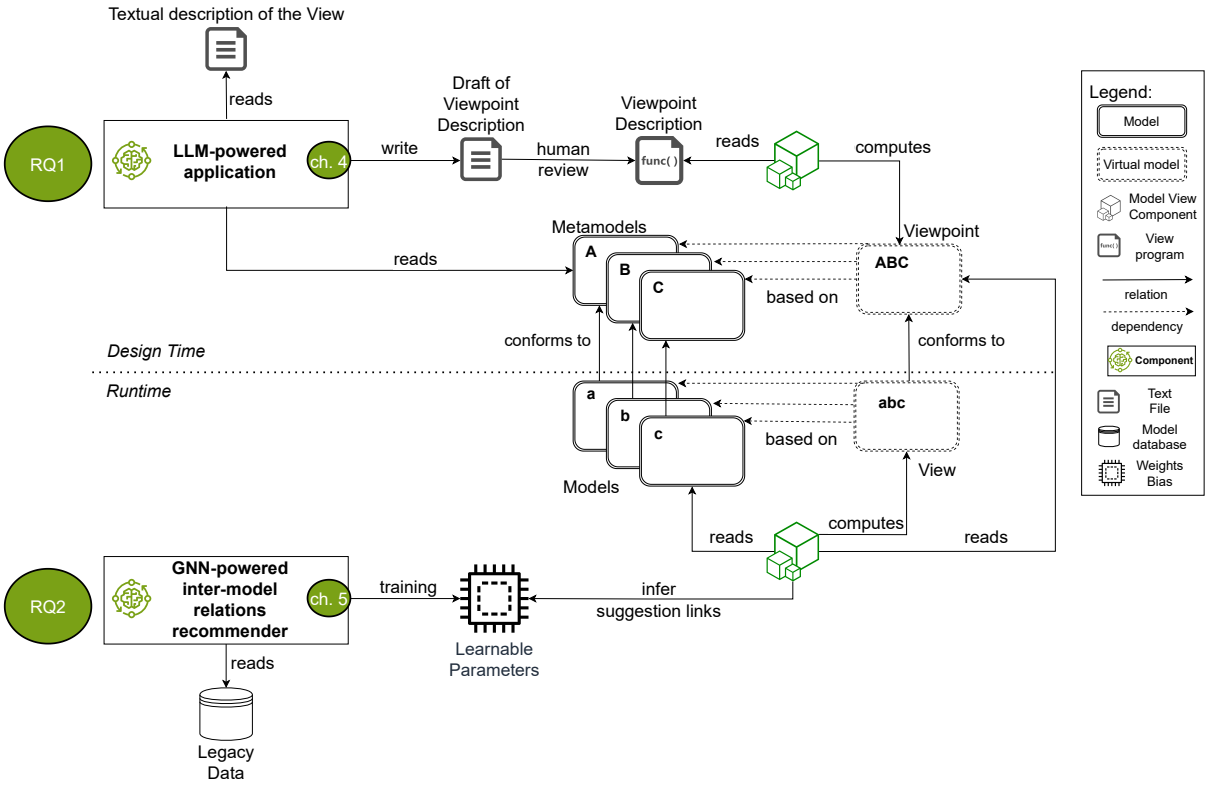

Figure 1.3 – An overview of the thesis contributions in an integrated framework

In this thesis, we contribute to answering the established research questions (cf. section 1.2) through the design and implementation of an ***enhanced model view solution powered by deep learning techniques***. To conduct this research, we have followed the

principles of the Design Science Research (DSR) [89]. This methodology is appropriate for our work since it emphasizes (i) the development and enhancement of purposeful IT artifacts and (ii) empirical validation through practical implementation.

For (i), our work proposed enhanced versions of modeling tools. More specifically, the work proposes using both LLMs and GNNs applied to the EMF Views tool.

For (ii), we provide empirical validation through practical implementation of the proposed solution in the context of the use cases provided by an industrial partner collaborating with the AIDOaRt project.

In summary, the six main activities of the DSR methodology are:

1. *Problem identification and motivation*: We investigated the MDE studies, with a focus on the model federation and mainly model views in search for the use of AI techniques applied to it. It helped us to have an overview of the research space, design our motivation, and find strengths and limitations. In addition, we also identified a suitable use case among all AIDOaRt partners for the use of model views.
2. *Objective definition for a solution*: We identified the requirements for providing DL-enhanced modeling tools for model views creation.
3. *Design and development of the artifact*: We detailed the approach of both how to use LLMs in the design phase of model views development and also how to use GNNs during runtime to compute these views effectively and with expressiveness.
4. *Demonstration through use cases*: We implemented both approaches for the ***EMF Views*** tool. We demonstrated them using examples gathered from the literature.
5. *Evaluation using predefined criteria*: We evaluated our solution empirically adapting the proposal of *Technical Risk & Efficacy* strategy from Venable *et al.* [90]. It means that for each contribution, we performed a formative evaluation at the end based on selected examples and metrics appropriate for each contribution.
6. *Communication of results*: The results of this thesis are published as described in section 1.5.

Following the established methodology, the thesis's primary contribution is enhancing pre-existing modeling tools by integrating DL techniques. Figure 1.3 provides a conceptual overview of the proposed approach. It illustrates two key DL-based components (driven by LLMs and GNNs) that collaborate to construct and refine model views within a multi-phase workflow, encompassing both the design time and runtime stages.

Specifically, LLMs assist view engineers during the design phase by enabling the definition of the view with minimal input. Additionally, GNNs aids ML-specialists in par-

ticipating in model views definition, identifying inter-model links, thereby improving the expressiveness and utility of model views.

The upper portion of Figure 1.3 is a high-level diagram illustrating how LLMs are leveraged during the design phase to draft the initial viewpoint description. This first draft is refined through human review and compiled into the final viewpoint. Since LLMs's natural language interface allows engineers of varying expertise levels to contribute by providing a *Textual description of the View* without necessarily deep knowledge of the underlying models. This input serves as the foundation for the first major contribution of the thesis: a *LLM-powered application* to aid the viewpoint description definition. Developing this component helps to answer the **RQ1**. The contribution is detailed in Chapter 4.

Once the design phase concludes, the challenge shifts to runtime, where effectively identifying inter-model links is critical. This challenge can be due to limitations in the query capabilities of the DSL used during the design phase or by engineers' limited familiarity with the underlying models. When sufficient *Legacy Modeling Data* is available, we can train a Neural Network (NN) to identify and recommend potential links, even when not explicitly expressed during the design phase.

The lower part of Figure 1.3 highlights the *GNN-powered inter-model relations recommender*, which applies GNNs for link prediction in model views. GNNs are particularly well-suited for this task due to their ability to handle graph-structured data, making them ideal for models in the MDE. Developing this component helps to answer the **RQ2**. The contribution is detailed in Chapter 5.

As stated before, we have implemented both contributions within the *EMF Views* plugin [64]. Following the established methodology, the enhanced EMF Views plugin is a significant IT artifact developed to meet the identified challenges. We use it through an industrial use case as a motivational example, demonstrating its practical application and benefits (cf. 2.3). When applied to the EMF Views, the overall analysis of these two contributions helps us to answer the **RQ3** since it allows us to verify its benefits and limitations.

1.4 Outline of the Thesis

We initially provide the background in the context of this thesis, together with a running example gathered from the AIDOaRt partner in Chapter 2. It is complemented with a state-of-the art on model view solutions and on the use of DL in MDE context in

Chapter 3, The contributions of this thesis are then presented in Chapters 4, and 5.

Chapter 4 presents the provided facilities for the use of LLMs in the EMF Views tool to help in the design of the view without deep knowledge of the involved metamodels. *Chapter 5* show the applicability of GNNs to find inter-model links when considering this problem as a link prediction problem.

In the end, we conclude the thesis in Chapter 6 with a discussion on the achieved results, recalling the research questions. Possible future research directions close the manuscript in Chapter 7.

1.5 Scientific Production

The outcome of this thesis has been published in one journal and three conferences.

— International journal

1. **J. Miranda**, H. Bruneliere, M. Tisi, and G. Sunyé, "Integrating the Support for Machine Learning of Inter-Model Relations in Model Views," The Journal of Object Technology, vol. 23, pp. 1–14, Jul. 2024, doi: 10.5381/jot.2024.23.3.a4.

— International conferences

1. **J. Pontes Miranda**, H. Bruneliere, M. Tisi, and G. Sunyé, "Towards the Integration Support for Machine Learning of Inter-Model Relations in Model Views," in Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing, Avila Spain: ACM, Apr. 2024, pp. 1304–1306. doi: 10.1145/3605098.3636143.
2. J. Cederbladh, L. Berardinelli, H. Bruneliere, A. Cicchetti, M. Dehghani, C. Di Sipio, **J. Miranda**, A. Rahimi, R. Rubei, J. Suryadevara, "Towards Automating Model-Based Systems Engineering in Industry - An Experience Report," in 2024 IEEE International Systems Conference (SysCon), Apr. 2024, pp. 1–8. doi: 10.1109/SysCon61195.2024.10553610.
3. **J. Pontes Miranda**, H. Bruneliere, M. Tisi, and G. Sunyé, "Towards an In-Context LLM-Based Approach for Automating the Definition of Model Views," in Proceedings of the 17th ACM SIGPLAN International Conference on Software Language Engineering (SLE '24), Pasadena, CA, USA, Oct. 2024, pp. 1-14.doi: 10.1145/3687997.3695650.

Chapter 2

Background

This chapter describes and presents the terminology and context of the main topics of the thesis, namely ***Model Views*** and ***Deep Learning***. It aims to cover the necessary background the reader needs to understand this thesis' contributions.

The section 2.1 starts with an introduction to MDE concepts in sub-section 2.1.1, including concepts around model transformations and weaving models. This helps to explain the model views concepts introduced and explained in sub-section 2.1.2. We also describe the tool of interest for this work that implements model views for EMF-compliant models, the *EMF Views*, in the sub-section 2.1.3.

In section 2.2, we introduce the main concepts around DL as a subset of ML. We start by introducing NNs in sub-section 2.2.1 that we considered essential for this thesis, focusing on their applications through the use of GNNs and LLMs in the subsections 2.2.2 and 2.2.3, respectively.

Finally, section 2.3 presents an industrial use case for EMF Views that motivates the work, and section 2.4 summarizes and concludes the chapter.

2.1 Model Views

Model Views are abstractions used during system development to describe a specific perspective over the system [19, 21, 59, 60]. They generally focus on selecting specific elements from the models and hiding irrelevant details during the system's analysis or execution. At the same time, a model view can also include new *virtual* elements, non-existing in the original models, which can also help represent the system from a specific viewpoint [21, 64].

In practice, model views are an instrument for handling the complexity of models, making it easier to understand, analyze, and manipulate the parts of the model that interest a particular stakeholder or for a specific task. This is a critical aspect of using model views, as it helps separate concerns, as different stakeholders or engineering tasks

may require different information.

This section starts with a brief introduction on MDE context to state the concepts and the terminology used in the rest of the thesis, followed by definitions around model views and the presentation of the EMF Views tool.

2.1.1 Context

The term "model" is among the most overused words in various scientific fields [94], from philosophy to mathematics, including different applications and different meanings [45, 95]. Essentially, we call a model a representation of an actual entity (e. g. an object, a system, a person, or a group) used to understand it better, simulate some situation, validate different assumptions, or make predictions about the matter [96].

In this thesis, we use the definition of models commonly shared in MDE community. A model is a set of abstract elements describing (part of) some system. It captures essential information while omitting unnecessary details [13, 97–100]. Generally, these model elements can be (semi-)formally defined (e. g. in [101]) or more broadly defined as postulates, and data presented visually in material form, in mathematical terms, or as a computer program that shares important characteristics with its real-world counterpart [96]. Models can be used from simple communication between stakeholders up to the actual model transformation in system implementation or other relevant artifacts (e. g. source code, documentation, and reports) [97] and also for analysis, estimation, simulation, and testing. Still, they are always intentionally created for some specific purpose, to execute some particular task [97].

We can primarily divide the different engineering approaches that deal with this kind of model into two big groups: Model-Driven-star (MD*) and Model-Based-star (MB*) approaches.

MD* refers to the approaches that use models as first-citizen artifacts and use them to drive indeed the rest of the engineering process [13], which means that creating, developing, and studying a given system is made through the models. MD* approaches emphasize automating development tasks through model transformations(cf. sub-section 2.1.1.3), code generation, and model validation to improve productivity and consistency [13]. Taking Software Engineering (SE) as an example, the Model-Driven Development (MDD) is a development paradigm that uses models as the primary artifacts for software development, automatizing the implementation and automatically generating as many as possible artifacts (e. g. code, documentation, logs, traces) from the models [97, 102]. Within Model-

Driven Software Engineering (MDSE) frameworks, we can go even beyond the generation of the implementation and use models to handle different SE activities [13], such as model-driven reverse engineering [103] and model-driven requirements engineering [104].

MB* approaches also use models to represent the system as first-citizen artifacts, playing a critical role in the engineering process [15]. MB* uses models to understand and manage system complexity, often as representations for analysis, simulation, and validation [15]. However, they are not necessarily used to generate the whole system's implementation and are frequently used as communication tools and blueprints [13]. MB* practices provide suitable ways of tackling the complexity of engineering CPSs [15, 16]. For the SE example, we can consider the Model-Based Software Development (MBSD), where models are used for different communication purposes and to create diagram sketches of the systems being developed [105].

In summary, Model-Based Engineering (MBE) is a broader concept encompassing all MB* approaches and treating models as central to system specification and verification. MDE plays a similar role encompassing MD* engineering activities where models are the key artifacts of the development and support their complete transformation into executable systems, aiming for an end-to-end model-driven process.

In the early 2000s, the Object Management Group (OMG™) established the approach of Model-Driven Architecture (MDA™) to provide users with tooling for model management to solve integration issues that complex systems could raise [106]. Towards that purpose, they define a unified approach to specify IT systems implementing the MDE/MBE through a set of well-defined standards [106].

Successful examples of engineering guided by models (either MDE or MBE) can be seen in a wide range of scientific and industrial domains [28, 29, 32, 66, 107–111]. Recent examples include models for blockchain and smart contracts [112], embedded systems [113] and e-government [114]. For this thesis, we used MDE as a general umbrella term. Still, our contributions also apply broader to MBE. Whenever the distinction is necessary to highlight some essential aspects of the system being developed, as in the AIDOaRt use case description (cf. section 2.3), we use more specific terminology.

The first generation of MDE tools primarily focused on code generation based on high-level abstract descriptions. Since then, the target scope of MDE has changed. It proved to be helpful in all stages of the systems' lifecycle: early design [115], modernization [116], and refactoring [117] being just a set of example uses for MDE techniques and tooling. Using models as purposeful abstractions of systems is also increasingly important for

modern industrial applications, e. g. in CPSs and digital twins [118–120].

2.1.1.1 Model-Driven Engineering Terminology and Standards

This section presents the main terminology and some standards for MDE processes together with illustrative examples. It helps us have a common vocabulary when discussing the rest of the thesis.

System Generally, a system refers to a real-world entity or artifact composed of regularly interacting or interdependent groups of items [121]. These systems can range from physical to software and include other engineered items. In the MDE context, the system is the target of analysis, design, and development, where models represent different aspects of the system's structure, behavior, or functionality [13, 97, 98].

Models Models are abstract representations of systems that serve various purposes, such as understanding, analyzing, or simulating the system before it is fully built or deployed [13]. Specific *modeling languages* are used to define and construct these models, and they can be of two types:

- General Purpose Languages (GPLs): languages that can be used to model a wide range of aspects of the same system. An example of this kind of language is the SysML™, used to model a broad range of systems and systems-of-systems. Systems modeled using SysML™ may include hardware, software, information, processes, personnel, and facilities [88].
- DSLs: languages tailored to a specific application domain, enabling experts to create a system using the concepts they are familiar with [122]. To cite a couple of examples, we have *Dsl4gar* as a DSL for modeling gaming rules to be applied to non-playful environments [123] and *SEMKIS-DSL* as a specific DSL to specify dataset requirements and expected skills for NNs [124]. The professional who defines a DSL is often called a *language engineer*, and the language user who defines models using the DSL is referred to as a *domain expert* [125].

As an illustrative example, we introduce two models in the following, both representing two different aspects of the same example system:

1. **Java model**: Figure 2.1 shows an object diagram that partially represents the structural hierarchy of a Java program. For this example, we choose to exemplify

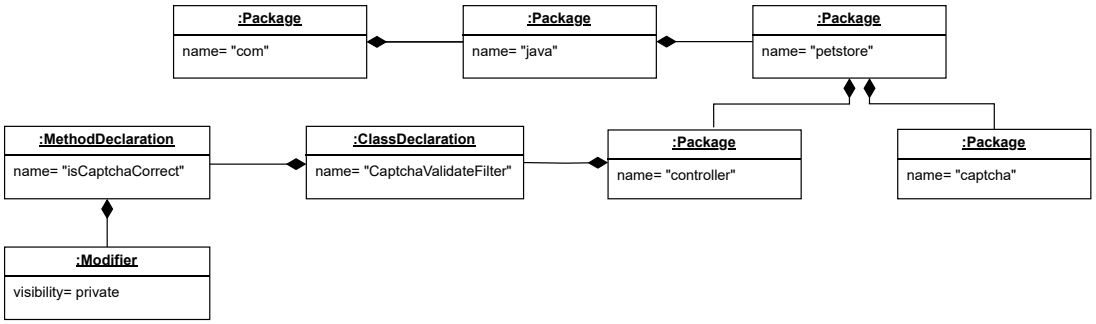

Figure 2.1 – Excerpt of a Java model for a pet store e-commerce application as an object diagram

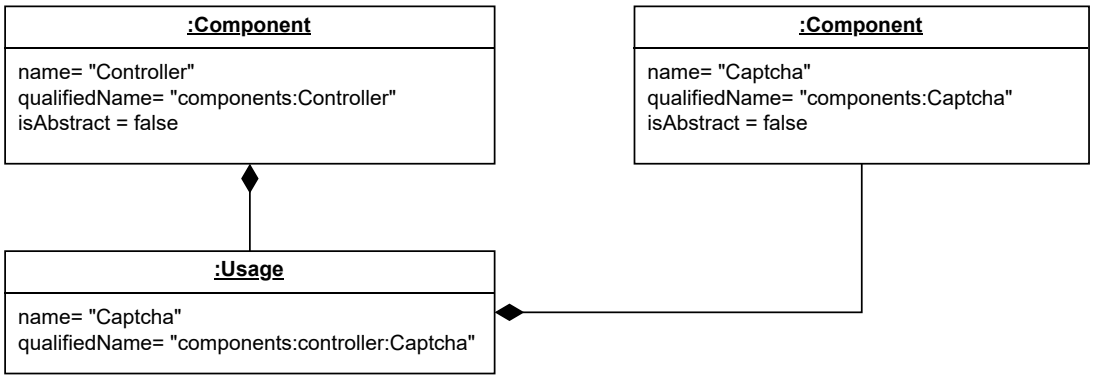

Figure 2.2 – Excerpt of a UML model for the captcha components in web application as an object diagram

it with a web application for a pet store e-commerce. [1] It includes three top-level packages: *com*, *java*, and *petstore*. The petstore package includes two sub-packages: *captcha* and *controller*. Within the controller package resides a class, *CaptchaValidateFilter*, which has a single method named *isCaptchaCorrect*. This method is marked with a *private* visibility modifier.

2. **UML model**: Figure 2.2 shows an object diagram represents a system's components and their dependencies, showcasing two concrete component objects: *Controller* (*components:Controller*) and *Captcha* (*components:Captcha*), both marked as non-abstract. The *Controller* component depends on the *Captcha* component, as represented by a *Usage* object. These components partially represent the same web application used for the Java model, i. e. a pet store e-commerce application. Still, it depicts it at a higher level (e. g. a captcha validator web component). [2]

Metamodels A metamodel is a model that defines the structure and constraints of other models, acting as a "model of model," specifying what elements models can contain and the relationships between them [13]. Formally, a metamodel is an explicit specification of an abstraction [126], strongly related to the concept of ontology [127]. Creating a metamodel consists of the definition of *concepts* (or *classes*) that contains *properties* of a certain *type*. Classes are related to each other through *relations*. In complement, *semantic rules* and *constraints* can also be included in the metamodel, such as multiplicities and containment references [13, 126, 128].

As examples of metamodels, we present in the sequence both metamodels of the two models given before, which means the Java metamodel and the UML metamodel.

Figure 2.3 shows an excerpt of the Java metamodel as a class diagram. [3] The Java metamodel defines all the entities that can be written in Java. From our previous example: `Packages`, `Classes`, `Methods`, and `Modifiers`. The UML metamodel illustrated in figure 2.4 [4] is adapted from the Eclipse MDT project. [5] This UML metamodel excerpt defines the

1. The presented illustrative example is simplified. The full version of this model is accessible at `http://bit.ly/4eHXDuY` *(Last Accessed in November 2024)*

2. Similarly, this example is also simplified. The full version of this UML model is accessible at `https://bit.ly/3OqHhw1` *(Last Accessed in November 2024)*

3. It is a simplified version of the metamodel from the MoDisco project. The complete diagram and complementary information are accessible at the MoDisco documentation page on `https://bit.ly/4fBQ7Dd` *(Last Accessed in November 2024).*

4. Similarly with the Java metamodel, the complete diagram is accessible in the official project repository at `https://bit.ly/3V23PY2` *(Last Accessed in November 2024)*

5. `https://projects.eclipse.org/projects/modeling.mdt.uml2` *(Last Accessed in November*

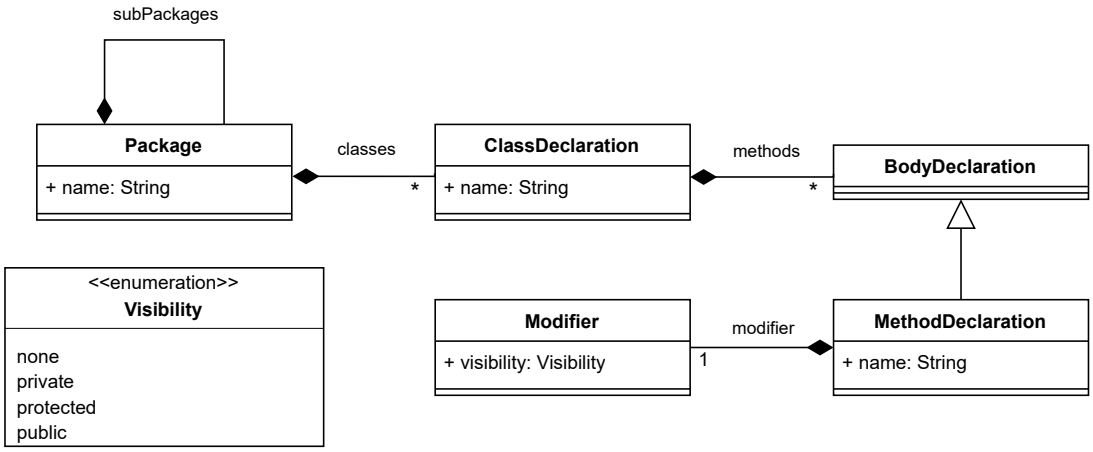

Figure 2.3 – Excerpt of a Java metamodel as a class diagram. It defines the structure of the model of Figure 2.1

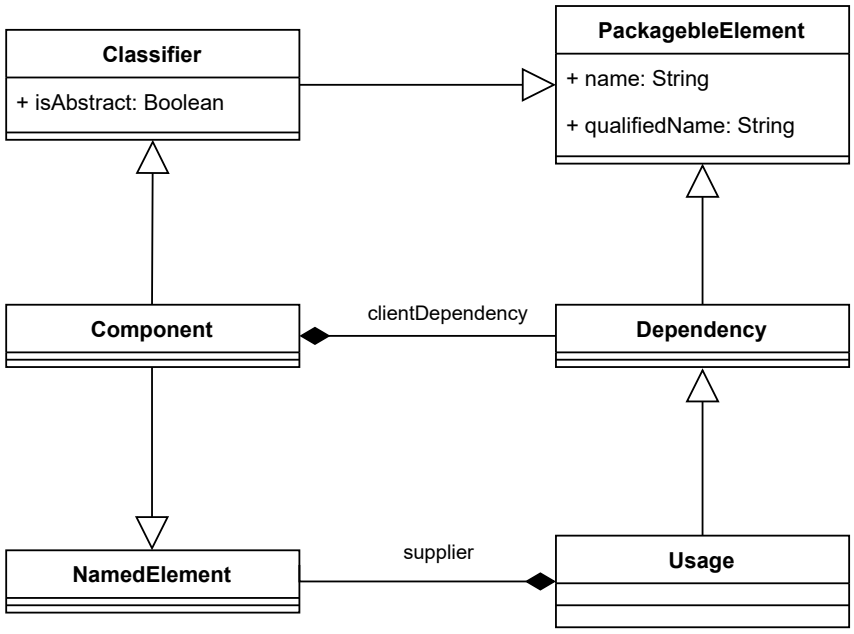

Figure 2.4 – Excerpt of a UML metamodel as a class diagram. It defines the structure of the model of Figure 2.2

relationships between components, dependencies, and usages in a given software system. `Component` inherits from `Classifier`, which includes the `isAbstract` attribute, and from `NamedElement`, which represents general named entities. Component instances can have a `clientDependency` relationship with a `Dependency instance`, which itself connects to a `Usage` instance through the `supplier` relationship. Both `Dependency` and `Usage` are specialized types of `PackageableElement`, which define common attributes like `name` and `qualifiedName`.

Metamodels define DSLs, ensuring they are well-structured in the formal modeling foundations. When defining a DSL through metamodeling, the produced metamodel is seen as the *abstract syntax* of the language [129]. For a given DSL, the abstract syntax is complemented by its corresponding *concrete syntax*, which specifies a representation for the elements to be used by the domain expert (e. g. providing graphical or textual symbols). In this thesis context, the term *language* (or *modeling language*) can often be used interchangeably with metamodel. Also, it is worth mentioning that the main focus of our contributions is on abstract syntax (i. e. metamodels, models) and does not consider the concrete syntaxes associated with the handled languages (either graphical or textual).

Model validation Model validation checks whether a model *conforms to* the rules defined in its metamodel. This is essential for ensuring that models are well-formed and meet the constraints imposed by the domain they represent. Metamodels usually govern language constraints, and additional arbitrary constraints can be specified for more intricate validation of model artifacts [128, 130, 131]. Advanced validation methods can be done through static analysis [132] and formal verification [133], for instance.

2.1.1.2 Modeling Languages

Among the languages for metamodel definition (i. e. meta-metamodel), we can highlight the OMG™'s Meta-Object Facility (MOF™),[6] which provides standard concepts for defining metamodels.

Developed by the Eclipse Modeling Project,[7] the *Ecore* is the meta-metamodel used to define the Ecore metamodel (i. e. Ecore is a self-defined metamodel, meaning that it defines its structure using the concepts it provides). Ecore is a practical implementation

2024)

6. `http://www.omg.org/mof/` *(Last Accessed in November 2024)*
7. `https://eclipse.dev/modeling/` *(Last Accessed in November 2024)*

of Essential MOF (EMOF) (simplified subset of MOF™) but tailored for the use and integration within Eclipse-based environments. It is at the core of the Eclipse Modeling Framework (EMF). EMF is a modeling and code generation framework that enables developers to define and automatically generate code for data models. All models and metamodels used along the thesis are EMF-compliant.

In addition to Ecore and other MOF™-based languages, there are some non-MOF™ alternatives like JetBrains Meta Programming System (MPS) [8] and the Microsoft Modeling SDK for Visual Studio. [9] MPS is a language workbench for defining custom DSLs using a projectional editing mechanism, interacting directly with the abstract syntax tree (AST). The Microsoft Modeling SDK offers MDE support in ".NET", similar to EMF, but designed for Microsoft technologies and development tools integration. Both technologies represent alternatives to MOF™-based languages. Although this thesis's contributions may be portable to them, the inner details of this portability are not discussed.

2.1.1.3 Model Transformations

Model transformations are a core concept in MDE, enabling the generation of one or more target models from one or more input models [13]. Model transformations are defined at the metamodel level, with no explicit references to their instances. They are then applied to models conforming to these metamodels.

While general-purpose programming languages like Java or Python can define model transformations, DSLs such as Atlas Transformation Language (ATL)[134] [10] and Epsilon Transformation Language (ETL)[135] [11] are commonly used due to their tailored features for this task. Usually, these transformations are expressed as a set of transformation rules. Transformation rules specify how elements in source models map to semantically equivalent elements in target models.

Particularly for this thesis context, the model transformations' ability to systematically convert one model to another forms a particular type of view: views concerning two existing models and connections between semantically equivalent elements from these models. It is straightforward why some model view solutions implement the view concept backed by transformations (cf. section 3.1 for details). We use model transformations as a complementary example in our LLM-application (cf. Chapter 4 and the upper part of

8. `https://www.jetbrains.com/mps/` *(Last Accessed in November 2024)*
9. `https://bit.ly/3ZhtAG6` *(Last Accessed in November 2024)*
10. `https://eclipse.dev/atl/` *(Last Accessed in November 2024)*
11. `https://eclipse.dev/epsilon/` *(Last Accessed in November 2024)*

Figure 1.3).

2.1.1.4 Weaving Models

In a standard MDE project, numerous models can be used and transformed between them. Since each model represents a different aspect of a system, different metamodels often describe them, which introduces complexity in maintaining consistency across them. Maintaining consistency across these models introduces complexity, and establishing relationships between them becomes necessary [51, 52]. *Weaving models* capture these relationships, dependencies, or mappings between elements from different models [53, 54]. They conform to a weaving metamodel, specifying the types of relationships that can be represented. Following the "everything is a model" principle in MDE [13], weaving models can also operate at the metamodel level, linking elements of different metamodels.

Weaving models serve as enablers for model transformations by defining the connections and mappings transformations rely on [136]. They can guide transformations to preserve relationships between models.

In multi-view system modeling [19, 59], weaving models integrate viewpoints into a coherent system model [59, 60]. Model view solutions, such as EMF Views (cf. 2.1.3), often rely on weaving models for managing viewpoints and views. Our contribution (cf. Chapter 5) aims to partially automate the generation of weaving models, as illustrated in the lower part of Figure 1.3.

2.1.1.5 Overview

Figure 2.5 presents a concise overview of all modeling concepts presented in this thesis, together with the role played by the model views. The figure highlights how MDE addresses engineering challenges along two dimensions: conceptualization (columns) and implementation (rows). MDE's core process flows from application models to running systems via model transformations, enabling model reuse across platforms. Realization relies on a specific platform within a domain, and models are defined by a modeling language (i. e. metamodels) governed by a metamodeling language (e. g. Ecore or MOF™-compliant languages). Transformations are carried out through transformation rules written in a transformation language. The system is built through a top-down approach in a standard pipeline, using prescriptive models to define scope and implementation. At the same time, abstraction works bottom-up to produce descriptive models of systems [13] and model

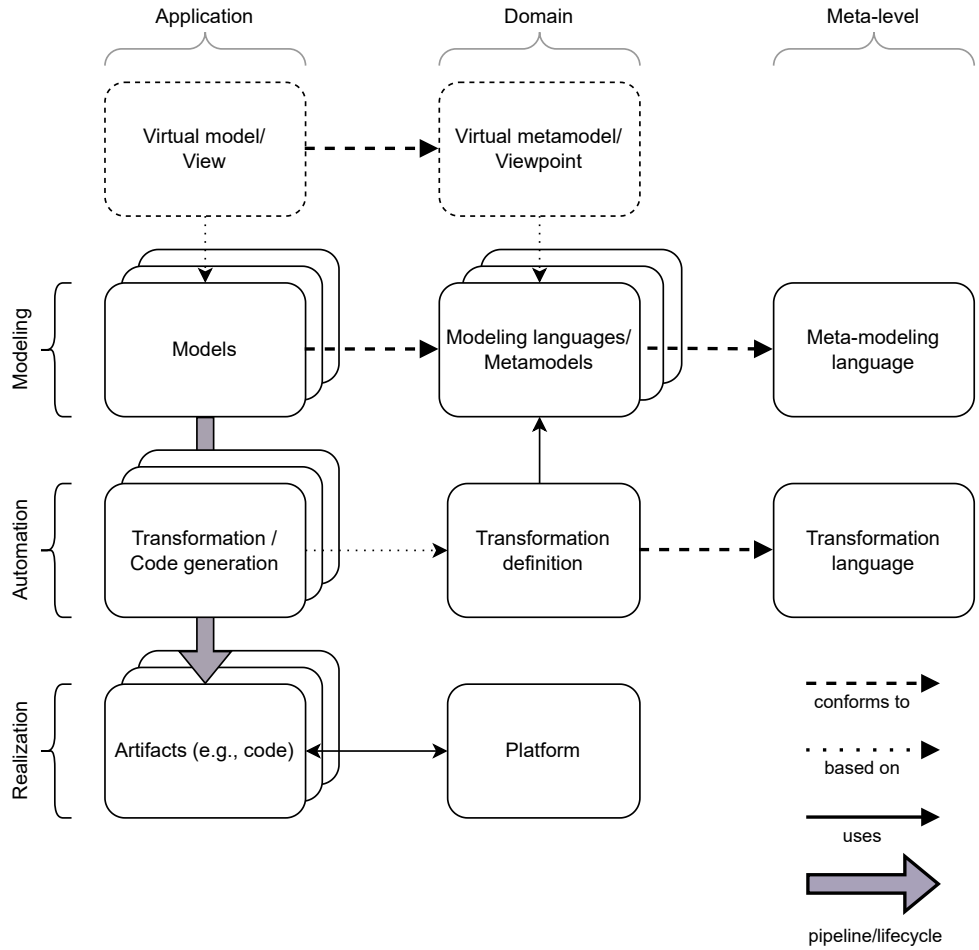

Figure 2.5 – Overview of modeling concepts and the role of view-modeling (Figure adapted from [13, p. 10])

views that can handle their complexity, abstracting specific perspectives and focusing on particular tasks.

2.1.2 Definitions and Example

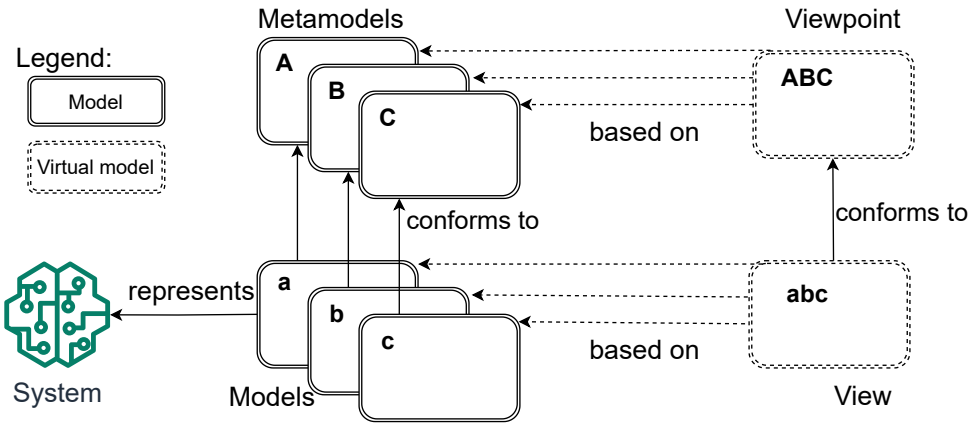

Figure 2.6 – Main concepts of model views

Essentially, a ***model view*** is a specific representation of a system, tailored to focus on a particular aspect, or *viewpoint*, depending on the stakeholders' concerns [21, 57, 59, 60]. It addresses the complexity of modern systems by separating concerns, which is challenging within MDE [37]. The system information is extracted from one or more **contributing models** (also called **base models**), which might be augmented with additional metadata or connections to represent specific needs. This separation is essential to managing the various dimensions of a system's design, analysis, and evolution in complex domains like CPSs or large-scale enterprise systems [21, 59, 60].

2.1.2.1 Model Views Concepts

View-based modeling dates back to the 1990s and became more prevalent with object-oriented frameworks and standards like UML. The industrial standard "ISO/IEC/IEEE 42010" [58] defines that the system's architectural descriptions should be organized around a set of views, each corresponding to a different stakeholder concern. There are two main methods for organizing these views:

1. **Synthetic Approach**: Multiple system views are created and later integrated, explicitly specifying relationships between elements in different views.

2. **Projective Approach**: Views are automatically computed from one or more contributing models, often through model transformations or queries. This approach is particularly beneficial in reducing manual effort and maintaining consistency across views.

Within MDE, projective model views are notably implemented through combinations of metamodeling, model transformations, and queries over models [21]. Figure 2.6 illustrates model views concepts and terminology within MDE context. A given system can be described by various models that potentially conform to different metamodels (i. e. expressed in different modeling languages). At the metamodel level, a *viewpoint* determines which concepts and properties from the contributing metamodels should be included or excluded in the corresponding views. It also expresses how these concepts should be interconnected, i. e. with which rules. At the model level, a *view* combines a given set of contributing models conforming to this viewpoint. The following provides a textual explanation of this terminology that will be used in the rest of the thesis.

Viewpoint A *Viewpoint* describes a combination, partitioning and/or restriction of concerns from which systems can be observed. It involves a collection of concepts coming from one or more metamodels, eventually complemented by new interconnections between them and newly added features.

View A *View* is a representation of a specific system from the perspective of a given viewpoint. It consists of a set of elements coming from one or more base models, eventually complemented with some interconnections and additional data, either manually entered or computed automatically (e. g. via one or more model transformations).

Virtual model A *Virtual Model* (respectively, metamodel) only points to elements from the original models (respectively, metamodels), thus preventing unnecessary information duplication.

2.1.2.2 Example

As a straightforward example of a model view, we describe a possible view connecting the Java and the UML models earlier described. We want to get a view that includes all the information of a Java `Package` and all information of the UML `Component` when they share the same name. This view is illustrated in Figure 2.7 as an object diagram.

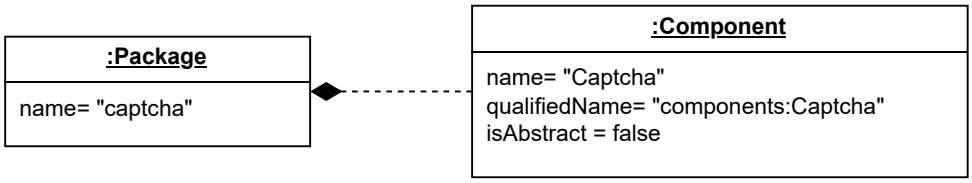

Figure 2.7 – View example connecting Java and UML with a virtual association.

Respectively, Figure 2.8 presents a simplified viewpoint responsible for defining the virtual association.

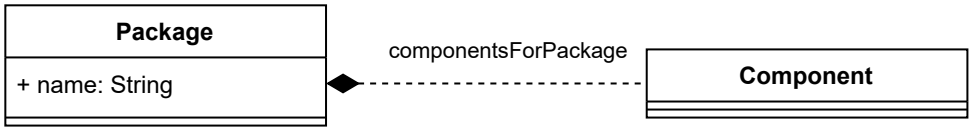

Figure 2.8 – Viewpoint example connecting Java and UML metamodels. The view illustrated in Figure 2.7 conforms to this viewpoint.

Being a view defined as a virtual model, it should be able to work as any other model. To illustrate it, Figure 2.9 shows an actual version of the example view that is created using the EMF Views tool (cf. section 2.1.3 for details). When opened in a standard model explorer (e. g. the MoDisco [12] explorer in the figure), the view should behave like any other model. Our previously presented diagrams omit some information from the original (meta)models to show them better in the manuscript. To create the actual view in EMF Views, we used the complete versions of their Ecore versions, which is why the explorer screenshot shows some information that was not mentioned before.

2.1.3 EMF Views solution

Our contributions currently target the EMF Views solution [42, 64]. The tool was first introduced by Bruneliere *et al.* [64]. The authors intended to provide a mechanism for combining models into cross-domain perspectives so stakeholders can have relevant views of the system being developed. Its development took place within the Naomod research group [13] and it is part of a MDE toolset [14] for dealing with complex engineering systems. A key motivation behind EMF Views was to create a generic, expressive, non-intrusive, interoperable, modifiable, and scalable solution. The work is a direct evolution of earlier

12. `https://eclipse.dev/MoDisco/` *(Last Accessed in November 2024)*
13. `https://naomod.github.io/` *(Last Accessed in November 2024)*
14. `https://www.atlanmod.org/` *(Last Accessed in November 2024)*

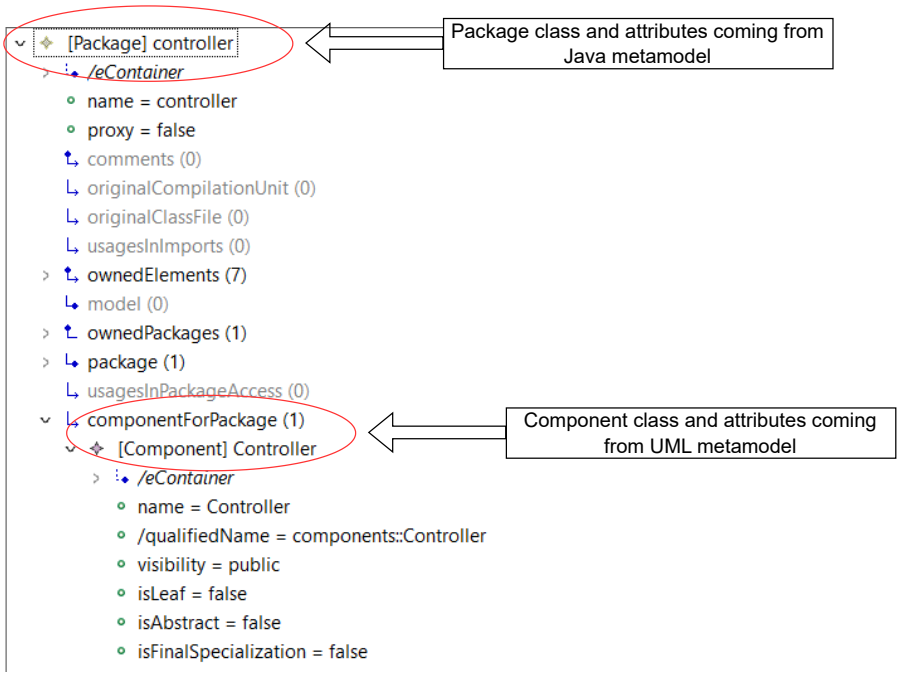

Figure 2.9 – Screnshoot of MoDisco model explorer presenting the example view.

attempts at model combinations using other strategies and mechanisms [52, 137]. EMF Views is distributed as an open-source Eclipse component. [15]

EMF Views follows the projective approach. Both viewpoints and views are manifested as virtual metamodels and models. Figure 2.10 illustrates its internal virtualization mechanism. A model view is composed mainly of elements proxied from the contributing models, with additional associations created between them that only exist in the view. Links between model elements are stored in a separate weaving model. This approach is non-intrusive and transparent, as it does not modify the original models and treats views like regular models.

EMF Views is built-in with the ViewPoint Definition Language (VPDL). Using the well-known SELECT-PROJECT-JOIN operators from relational algebra, the VPDL was inspired by database views. Formally, VPDL is a textual Structured Query Language (SQL)-like DSL for writing model view definitions. A VPDL file expresses a viewpoint, i. e. it defines the concepts and properties from the contributing metamodels that need to be selected, associated, or queried (and how). Then, EMF Views take this file as input to build corresponding model views on given sets of contributing models. When working in

15. `https://www.atlanmod.org/emfviews/` *(Last Accessed in November 2024)*

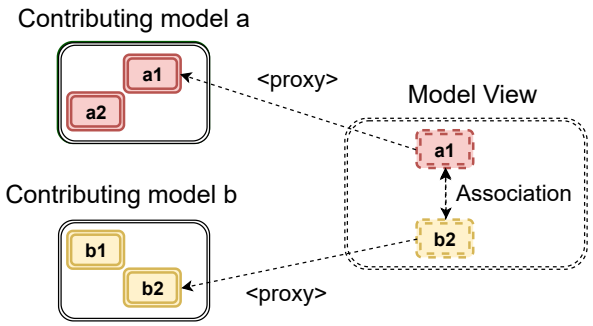

Figure 2.10 – High-level illustration of EMF Views virtualization mechanisms

the EMF environment, the EMF Views provide the necessary tooling to combine existing models in a single visualization.

```
1  create view chain as
2
3  select java.Model.*,
4         java.Package.*,
5         UML.Component.*,
6         java.Package join UML.Component as componentForPackage,
7
8  from 'http://www.eclipse.org/MoDisco/Java/0.2.incubation/java' as java,
9      'http://www.eclipse.org/uml2/5.0.0/UML' as UML,
10
11 where s.name = t.name.toLowerCase() for componentForPackage
```

Listing 2.1 – Example of a VPDL file combining a Java model and UML model

Listing 2.1 shows an example of a VPDL file that creates a viewpoint (and corresponding view) over our previously presented example metamodels, namely the Java metamodel and a UML metamodel. This VPDL file is essentially what we need to develop our example view (cf. section 2.1.2.2) within EMF Views. The code in the listing 2.1 contains all the elementary elements for a VPDL file. The line `1` is where the name of the view is defined. The name in this illustrative example is "chain". Lines `3-5` show the SELECT block, where the engineer can define which elements of each contributing model will appear in the final view. In our example, the metaclass `Package` is selected from the Java metamodel, and the metaclass `Component` comes from the UML metamodel. The character `*` means we want to select all attributes for the given metaclass. The line `6` is the JOIN block of the language and defines the combination between the two selected classes, giving it a mean-

ingful name as `componentForPackage`. The lines `8` and `9` define the involved metamodels, given by their Uniform Resource Identifiers (URIs). Finally, the line `11` defines how the combination specified by the JOIN will be computed in the view. Following our example, we are comparing the name of the Java `Package` and the UML `Component` so that each `Package` will have a potential associated `Component`. The VPDL code is responsible for creating the internal weaving model(s) as described in details in the sub-section 2.1.3.1 The tool documentation [16] shows the information on how to use VPDL and run the given example. This example was adapted from one of the EMF Views tutorials. [17] The screenshot previously presented in Figure 2.9 presents the actual computed view in the MoDisco model explorer.

2.1.3.1 Standard View Definition within EMF Views

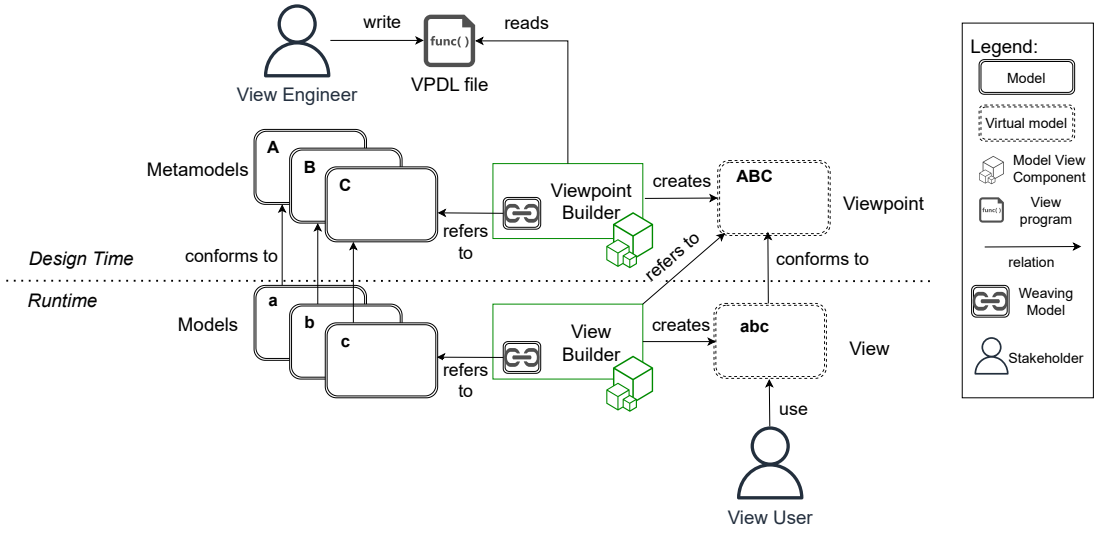

Figure 2.11 – Detailed overview of a view creation in EMF Views. Figure adapted from [138, p. 100]

Figure 2.11 details how this two-step approach works within EMF Views. It provides an overview of the design and runtime stages for creating viewpoints and views of a set of (meta)models. A `View Engineer` is responsible for creating a VPDL [18] file and defining the viewpoint. It has to be done after a careful analysis of the requirements and study of

16. `https://www.atlanmod.org/emfviews/manual/user.html` *(Last Accessed in November 2024)*
17. `https://github.com/atlanmod/emfviews/tree/master/examples/traceability-demo` *(Last Accessed in November 2024)*
18. EMF Views support other manners to create it, but we will focus on VPDL

the underlying models, often with the participation of a domain expert. At `Design Time`, the `Viewpoint Builder` uses the metamodels [19] as a reference for building a `Viewpoint` using virtualization. The `Weaving Model` plays a central role in this process by proxying parts of the metamodels to the `Viewpoint` (cf. Figure 2.10 for details). At `Runtime` the `View Builder` operates on the models to (semi-)automatically generate a specific view. [20] The `Weaving Model` again enables the selection and linkage of relevant elements and their computation. A `View User` interacts with the resulting views at runtime.

Since the building mechanism is used at both levels, a weaving metamodel was developed to describe the relations between the elements (both metamodel elements and model elements). Figure 2.12 shows the EMF Views weaving metamodel described in the following. [21]

WeavingModel The `WeavingModel` serves as the root element. It contains two key elements: the contributing models and virtual links. Virtual links represent modifications made to the models that only appear in the view, not the actual models. The `WeavingModel` includes a `whitelist` flag that changes how filters work. It indicates whether the view includes all elements from contributing models or only the specified ones. By default, the view includes all elements (i. e. "whitelist = true").

ContributingModel A `ContributingModel` is any model included in the view. It holds the actual elements targeted by virtual links. The `URI` attribute always refers to the metamodel's namespace URI for viewpoints and the view's weaving models.

ConcreteElement A `ConcreteElement` is an element within a contributing model. For viewpoints, its `path` attribute is the fully qualified name of the element (excluding the metamodel name, which is defined by the `ContributingModel` container). For views, the path is the URI returned by the EMF Application Programming Interface (API) method `"Resource.getURIFragment"`. Concrete elements have two subtypes: `ConcreteConcept` and `ConcreteAssociation`. This distinction is important because virtual associations, for example, can only connect associations, not any element.

19. e. g. the Java metamodel and UML metamodel from the example view. Represented as "A", "B" and "C" in the figure

20. e. g. the Java and UML models for the pet-store e-commerce example. They are represented in the figure as "a", "b" and "c"

21. A complete Ecore version of the weaving metamodel can be obtained at `https://bit.ly/3YWigOR` *(Last Accessed in November 2024)*

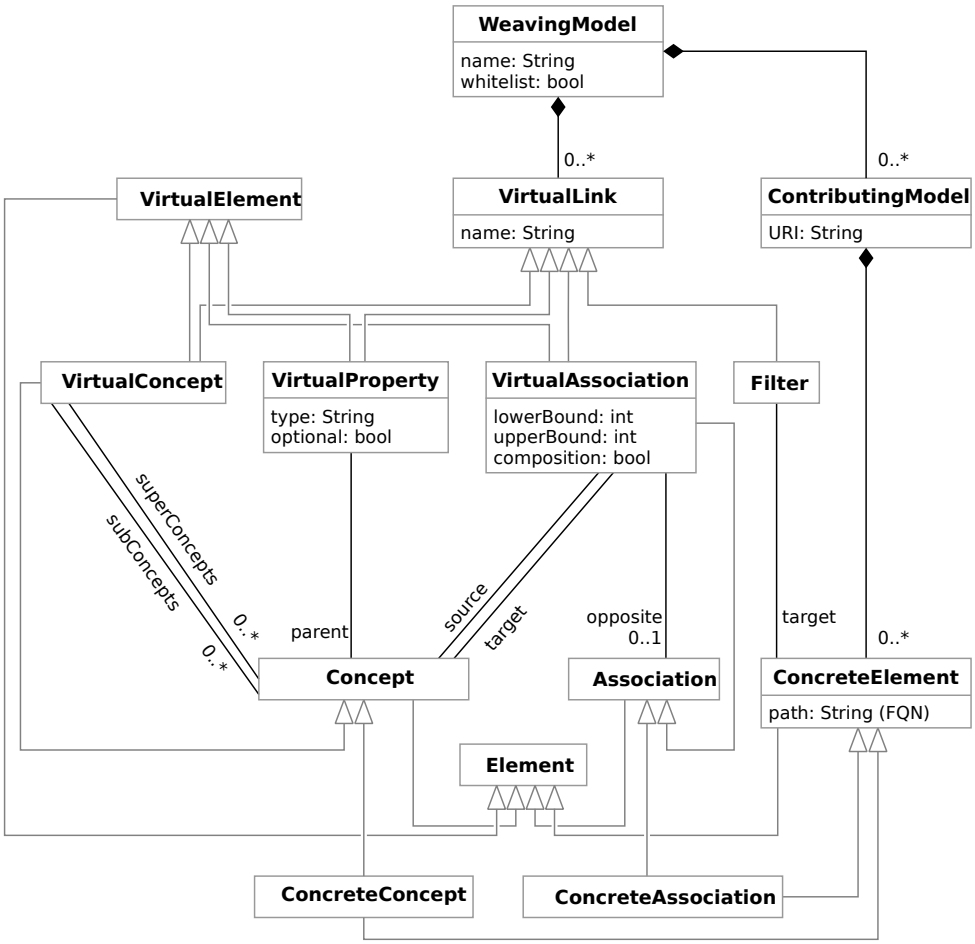

Figure 2.12 – Excerpt of the metamodel of weaving model as a class diagram. Figure from [138, p. 103]

VirtualLink `VirtualLink` is a parent class for all modifications to the model. Every modification has a `name` attribute, which becomes the name of the virtual feature—except for the `Filter` class, where the name is ignored.

Filter A `Filter` determines whether elements from contributing models are included or excluded, depending on the `WeavingModel`'s whitelist flag. `Filters` can only refer to `ConcreteElements`.

VirtualAssociation A `VirtualAssociation` is a relationship that exists only in the view. It connects a `source` and a `target`, either concrete or virtual concepts. This allows the creation of an association between a class from the contributing metamodel and a virtual class that only exists in the view. The `lowerBound` and `upperBound` properties set the association's cardinality, similar to how it works in Ecore metamodels. If the `composition` flag is true, the virtual association is a containment. A virtual association can have one opposite association (virtual or not), defined by the `opposite` reference.

VirtualConcept A `VirtualConcept` exists only in the view. It can be a `subclass` or `superclass` of other concepts, whether virtual or not.

VirtualProperty A `VirtualProperty` is a property that exists only in the view. It must be linked to a parent concept, either virtual or real. Its `optional` flag determines the property's cardinality. The `type` attribute defines the property's primitive type.

2.2 Deep Learning

In today's digital transformation landscape, AI technology applications are ubiquitous in many businesses and industries [67]. Since the middle of the XX century, it has evolved, and it has impacted different aspects of how we deal with our data and systems [10, 24, 67].

ML can be described as the use of well-crafted algorithms, often based on statistical techniques, that enable the computer to learn some patterns in the data and then generalize it to unseen data [22]. The application of ML can vary from a good guesser that can help build recommendation systems to precise predictors for precision-dependant systems. In summary, ML is a broad field of AI where algorithms learn from data to make

predictions without being explicitly programmed. It includes techniques like linear regression [139], decision trees [140], and Support Vector Machines (SVM) [141] to mention some examples.

DL is a class of ML algorithms that has gained attention and much traction in the last decade [142], assuming the role of the main ML technique for a wide range of applications. It focuses on using NNs, especially the ones with many layers, also called *deep neural networks*. While all DL is ML, not all ML involves DL, even with some overlap in the terminology of both study fields.

For this background chapter of the thesis, we decided to focus primarily on DL since it is foundational for the two ML applications we investigate in our contributions, GNNs and LLMs. The sub-section 2.2.1 introduces the main concepts around the development and use of NNs, while the following sub-sections 2.2.2 and 2.2.3 presents respectively the background and the current landscape for the application of GNNs and LLMs.

2.2.1 Neural Networks

For a fundamental definition, we can state that any ML algorithm operates on the same principle. Given a task T evaluated by some performance measure P, the goal is to improve P on T based on a set of experiences E [143]. In other words, ML algorithms are designed to enhance their performance on a specific task by learning from past experiences. A predefined criterion measures the algorithm's success, and the objective is to maximize this performance as it gains more experience [143].

Deep learning is a class of algorithms used to learn representations of data (i. e. embeddings) using multiple computational layers, allowing multiple abstraction levels. It can discover intricate structures in large datasets, indicating how a machine should change its internal parameters, which are used in the next layer to re-compute the data representation, achieving some understanding of the analyzed data [25]. The composition of layers creates a network [25]. Deep networks have been developed in recent years with breakthrough results in many areas, from speech recognition to genomics [27], and also with applications in software engineering [68]. In addition, the introduction of transfer learning [144] and pre-trained models such as transformers has significantly accelerated progress in numerous fields, including Natural Language Processing (NLP) and vision tasks.

ANN (usually referred to simply as Neural Networks (NNs) in the Computer Science

(CS) context) are the main algorithm [22] for DL. Because of their ability to reproduce and model non-linear processes, NNs can be used on a wide range of applications, from data classification to content generation (i. e. any task T). What we generalized as the experience E from the fundamental definition can be called in the NN context as the *training data*, which includes both a set of the input data (i. e. the data we intend to use as the source of the process) and the set of *output data* or *examples* (i. e. the expected outcome of the process the network is learning). The network's success is measured by how accurately it performs the task T compared to the training data. Depending on the nature of T, different metrics are often used to evaluate the network's performance P (e. g. accuracy, precision, recall, or more task-specific metrics like Intersection over Union (IoU) for object detection [145]).

Conceptually, a NN [23] loosely tries to mimic a biological brain, which means that its minor units (usually called *neurons*) are connected by *edges* that pass some information between them in a simulation of the process of a brain synapse. This mimicking process is achieved with artificial neurons that receive some input, perform a weighted operation on it, and then pass their results throughout the network to other neurons through a function. It is important to note that the analogy with a biological brain is just a conceptual approximation, and nowadays, it works to explain the concepts around NNs and not to develop this kind of system.

Technically, each neuron in a NN receives a *signal input* and sends the *output* of a *non-linear function* to the next neuron. Each signal is a real number, and the strength of the signal is given by its *weight*. In practical uses, the neurons are organized in groups forming *layers* of neurons, the input signal is a *vector* of real numbers, and the output is given by applying the *activation function*, such as the commonly used ReLU [146]. The following layer receives some aggregation (e. g. sum) of the outputs as its inputs. NNs with two or more layers are called deep neural networks [22, 25]. Advanced optimizers like Adam [147] and techniques like batch normalization [148] are now standard practices to face issues such as vanishing gradients in deep networks.

The training of a NN starts with the definition of a *loss function* with a paired *cost function*. Learning occurs by executing the pass of a set of inputs throughout all the layers

22. The ML community usually uses the word "models" to refer to the algorithms. To avoid confusion with the definition of models on MDE scenario, we preferred to use "algorithms" instead everywhere possible, excluding the cases where the distinction is noticeable. Different NN structures are referred to as architectures

23. from now on, ANN and NN will be used interchangeably, with preference for the second term.

of the NN and adjusting the weights to minimize the loss function. The primary algorithm used to minimize it is the *gradient descent* through *backpropagation* [149]. Backpropagation aims to update the weights so that the NN makes better predictions. *Supervised learning* is the type of learning where the NN receives both the inputs and expected outputs during the training. The loss function is defined as the deviation of the actual value (i. e. the training data) from the predicted value (i. e. the output of the network).

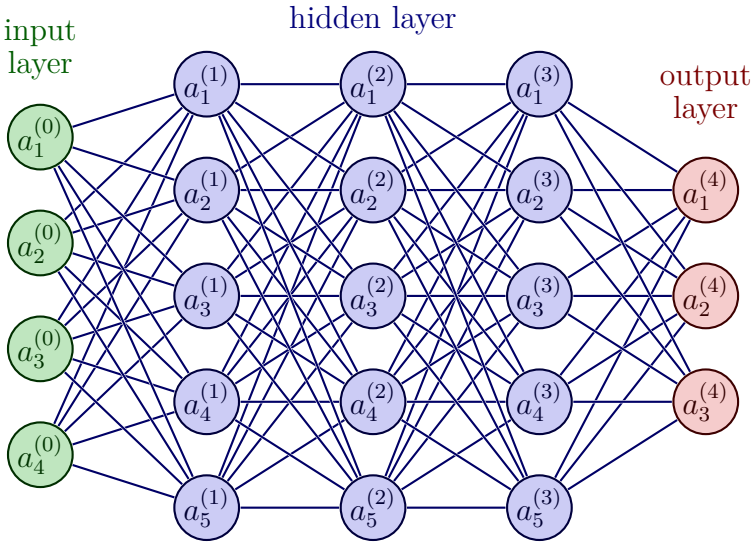

Figure 2.13 – Example of a fully connected feed-forward neural network [24]

NNs have evolved into a broad family of algorithms and were applied in a vast number of applications in different domains. The variations of each type of NN can be called NN architectures, and they are defined by adjustments to the network *topology* and its *hyperparameters*. The most straightforward architectures have static components such as the number of neurons (i. e. units), number of layers, neurons' weights, and standard feed-forward topology. Some other types can have dynamic components that evolve during training, so specific algorithms are also necessary to guide these changes [150]. Another significant variation that affects the performance of a NN is the type of hardware used for the training/inference process, mainly when switching from Central processing units (CPUs) to Graphics processing units (GPUs), and even further with hardware accelerators like Tensor processing units (TPUs) [151], which are optimized for large-scale computation.

24. Example figure adapted from `https://tikz.net/neural_networks/` *(Last Accessed in November 2024)*

Figure 2.13 illustrates an example of a fully connected feed-forward neural network composed of five layers of neurons. The first layer is the **input layer**, labeled as $a^{(0)}$, where each neuron $a_i^{(0)}$ represents an input feature of the data (i. e. the superscript indicates the number of the layer and the subscript indicate the index of the neuron on the given layer). In this example, the input layer has four neurons corresponding to four different input features (i. e. input signals). The middle section represents the **hidden layers**. In this example, there are three hidden layers, each one with five neurons. The layer on the right side is the **output layer**, labeled as $a^{(3)}$. This layer contains three neurons representing a task's possible output or class label. In a fully connected neural network like this, every neuron in each layer is connected to all neurons in the subsequent layer. The lines between the neurons represent these connections, and the strength of these connections is determined by the **weights** of the network (not presented in the figure), which are learned during training. This type of architecture is typically used for tasks like classification or regression, where the network processes the input features through the hidden layers to produce a set of outputs.

Varying the architecture of the NN has given us some particular types of networks widely used in various applications. In the following, we provide a non-exhaustive summary of the most well-known types, mainly when related to the two main NN types we use in the thesis, LLMs and GNNs.

Convolutional Neural Networks (CNNs) are designed explicitly for processing spatial data, such as images. Unlike feed-forward networks, CNNs use *convolutional* layers to scan and identify local patterns within the input using grid-like strategies to select parts of the input data [152]. This makes them more efficient for image recognition, object detection, and other computer vision tasks where spatial information is crucial. The essential pieces and strategies used in CNNs also inspired the development of GNNs, which extend the convolution operation to irregular domains like graphs (cf. sub-section 2.2.2 for details about GNNs).

RNNs differ from feed-forward neural networks as they have built-in memory, allowing them to process data sequences. They are well-suited for tasks like NLP and time series prediction. They can learn patterns in sequences by connecting the output from one time step to the input of the next, remembering previous information (the recurrence in the namesake) [153]. RNNs form the backbone of the *transformer* architecture together with the Long/Short Term Memorys (LSTMs). With its self-attention mechanisms, the transformer architecture is the base of the most used LLMs [154].

LSTMs are an improved form of RNNs, designed to better capture long-range dependencies in sequential data [153]. They do so by incorporating a set of gates (input, forget, and output gates) that control the flow of information, enabling the model to "remember" or "forget" specific parts of the data [155]. This capability makes LSTMs particularly effective for tasks involving long-term dependencies, such as time-series forecasting, language modeling, and machine translation [156].

Generative Adversarial Networks (GANs) introduced by Goodfellow *et al.* [157], are a different class of neural networks that consist of two competing networks: a *generator* and a *discriminator*. The generator aims to create data similar to a given training set, while the discriminator tries to distinguish between the real data and the data generated by the generator. Through this adversarial process, both networks improve over time, and GANs have been used in various applications such as image synthesis, style transfer, and even drug discovery [158].

The most transformative architecture in recent years has been the *transformer*, introduced by Vaswani *et al.* [154]. Transformers have become the foundation for LLMs such as BERT [159] and GPT [160]. Unlike RNNs and LSTM, transformers rely on *self-attention mechanisms*, which allow them to model dependencies between distant elements in a sequence without the need for recurrence. This parallelization makes transformers highly efficient for large-scale data processing, and their ability to capture intricate relationships in the data has led to state-of-the-art results in various domains, including NLP, code synthesis and even system design [161]. Given their versatility, transformers have become a dominant architecture for LLMs, a core technology explored in this thesis.

While many other NN architectures have been developed, such as Autoencoders [162] and Boltzmann Machines [163], these are outside the scope of this thesis.

Figure 2.14 loosely illustrates an evolution of NNs, highlighting the main architectures considered in the thesis. The basic architectures (e. g. feed-forward) were initially present by McCulloch and Pitts [164]. The development of the backpropagation algorithm for training neural networks allows the creation of architectures appropriate for working with more intricate data types. LeNet was the structure that introduced the concepts of CNNs [165], which had a breakthrough moment later with AlexNet [152] and ResNet [166], bringing high-level performance to deal with grid-like data (e. g. images). In a parallel branch of evolution, we had the proposal of GNNs [77] adapted to work with graph data. Inspired by the CNN architecture and the convolutional approach, the Graph Convolutional Network (GCN) extended the concept to enable better performance when

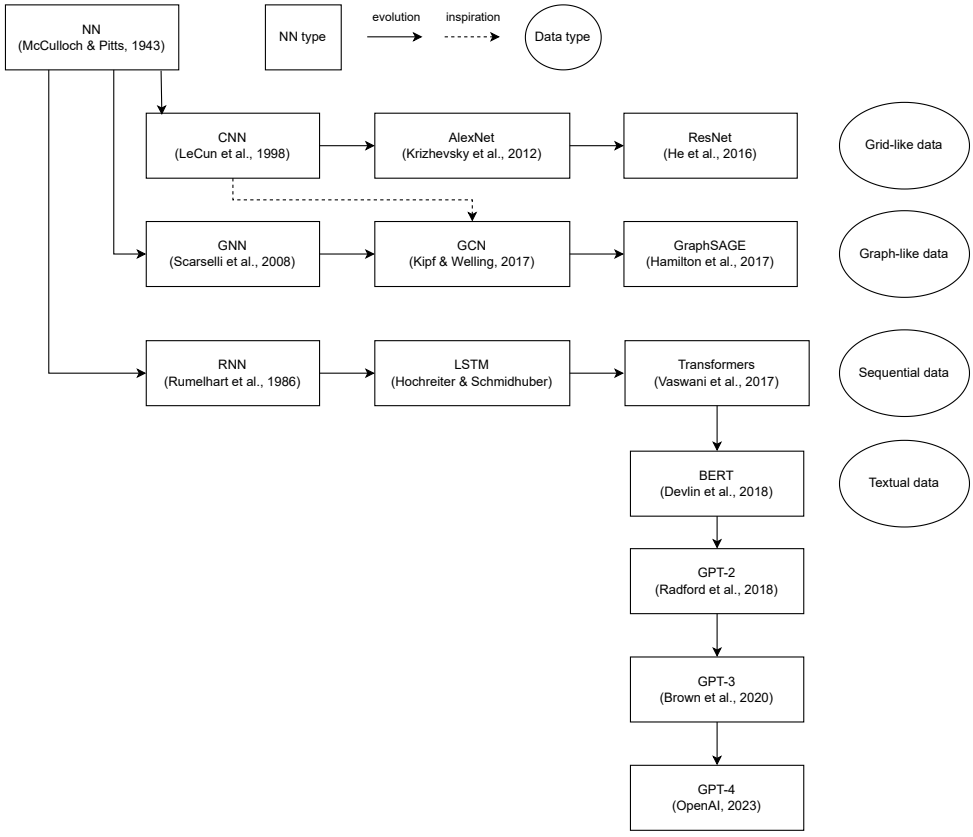

Figure 2.14 – Loose illustration of NNs evolution along time

dealing with graph-like data. The advances in graph data and its consequent increase in different applications have led to a new branch of research, creating various architectures. We highlighted the Graph Sample and Aggregate (GraphSAGE) [167] on the figure since we use it in our contribution (cf. section 2.2.2.2). As mentioned before, the origin of LLMs is somehow attached to the general evolution of architectures that deal with sequential data, like RNNs and LSTMs. The combination of them with the attention mechanism developed by Vaswani *et al.* enabled the development of the first language models like BERT [159]. The Generative Pre-trained Transformer (GPT) architecture evolved from it, creating a family of LLMs, from GPT-2 [168] to GPT-3 [160] and finally GPT-4 [169]. [25]

2.2.2 Graph Neural Networks

Natural phenomena, complex systems, and datasets can be represented as graphs. These are mathematical structures composed of objects, often referred to as *nodes*, that are sometimes connected by relationships known as *edges*. Graphs are also commonly referred to as *networks* and play a crucial role in the study of *discrete mathematics* and *graph theory*. Examples include social networks, citation networks in academic research, and interactions within biological systems such as protein-protein interaction networks [170].

In the context of graphs, the primary components are:

— *Nodes* (*vertices*) representing individual entities.

— *Edges* (*links*) that represent relationships or interactions between nodes.

The formal definition of a graph used in this thesis is as follows [171].

Definition. *A graph $G = (V, E)$ consists of a set of nodes V connected by edges in the set E, where each node $v \in V$ and each edge $e \in E$.*

Graph elements like nodes and edges can have associated features (e. g. labels or attributes). These are often represented with mapping functions $\phi(v) : V \to A$ and $\varphi(e) : E \to R$, where A and R denote node and edge features, respectively.

Graphs can be further classified based on the types of nodes and edges, leading to two major categories:

Definition. *A* homogeneous graph *is a graph where all nodes and edges have the same type, i.e., $|A| = |R| = 1$.*

25. Internal details of GPT-4 are not fully disclosed by OpenAI until the date of this thesis writing, but it is assumed to be a direct evolution from GPT-3

Definition. *A* heterogeneous graph *is a graph where multiple types of nodes and edges exist, i.e.,* $|A| + |R| > 2$.

The ability to represent both simple and complex relationships makes graphs a powerful tool for working with real-world data. Classical algorithms for graph-based problems, such as PageRank [172], have been widely used before the advent of DL approaches, providing foundational methods for analyzing graph structures.

Some common graph-based tasks include:

— **Node Classification**: Predicting the label of a node based on its attributes and the graph structure, for instance, identifying communities in social networks [173].

— **Graph Classification**: Classifying entire graphs is an essential task in areas such as molecular chemistry, where the graph structure of a molecule determines its properties [174].

— **Node Clustering**: Grouping similar nodes, often applied to wireless sensor networks for efficient organization [175].

— **Link Prediction**: Predicting missing or future edges between nodes, useful in fields like criminal network analysis to discover hidden connections [176].

— **Influence Maximization**: Finding the most influential nodes in a network, particularly for marketing in social networks [177].

However, classical algorithms for these tasks often struggle with generalization, scalability, and adapting to complex, heterogeneous data. The rise of *node representation learning* and GNNs have addressed many of these limitations.

Node representation learning, or *graph embedding*, is the process of transforming nodes into fixed-size vector representations, capturing the structure and features of the graph. This representation can be used for various downstream tasks, including the classical ones mentioned earlier. Traditional methods like DeepWalk [178] and node2vec [179] pioneered this field.

GNNs are a class of NNs designed to directly process data represented as graphs [77]. Unlike traditional NNs, suited for structured data like images or sequences, GNNs excel in tasks involving graph-structured data, where the node relationships are crucial.

These models have demonstrated notable success in various domains, such as social network analysis, molecule property prediction, and knowledge graph completion [180, 181]. Given the inherent graph-like structure of many problems in software engineering (e. g. dependency graphs, control flow graphs), GNNs are promising for applications such as program analysis, software fault localization, and code summarization [182, 183].

GNNs addresses the shortcomings of traditional methods by learning node representations through an iterative process of message passing and aggregation. Information is propagated between nodes based on their connectivity during message passing, allowing GNNs to capture both local and global graph structure [77]. Popular architectures include the GCN [180], which simplifies graph learning by using convolution-like operations, and the Graph Attention Network (GAT) [184], which incorporates attention mechanisms to weigh the importance of neighboring nodes.

2.2.2.1 Transductive vs. Inductive Learning in GNNs

One of the key distinctions in GNNs is between transductive and inductive learning paradigms. [26] In transductive learning, the model learns from a fixed graph during training, meaning that node embeddings are computed for the specific nodes and edges in the training graph. Consequently, this approach requires re-training if new nodes or edges are introduced, as the model has not learned a generalizable function that can infer embeddings for unseen parts of the graph [180]. In contrast, inductive learning allows the model to generalize to unseen graphs or nodes, as it learns a function that can compute embeddings for any node based on its local neighborhood structure and features. Inductive learning is useful when working with dynamic graphs, where the structure may evolve, or when the goal is to predict the properties of entirely new graphs.

2.2.2.2 Heterogeneous Graph Neural Networks (HGNNs)

Heterogeneous Graph Neural Networkss (HGNNs) extend the capabilities of standard GNNs to handle heterogeneous graphs. These models consider the semantics of node and edge types during message passing, resulting in more meaningful node representations for complex networks, such as citation networks or knowledge graphs. The message passing process in HGNNs is often based on meta-paths or relation-specific mechanisms, where messages are propagated along specific types of edges, capturing richer contextual information.

While GNNs and HGNNs have made significant strides, challenges remain, such as scaling to large graphs and learning robust representations in noisy or incomplete data environments [181].

26. This difference can be generalized to other ML algorithms. Still, we focused only on GNNs for simplicity

An important GNN architecture for the context of this thesis is the GraphSAGE, the example architecture used in one of our contributions(cf. chapter 5). We use it below as an example of architecture to explain part of the inner details of GNNs.

Formally, GraphSAGE is an inductive framework for learning node embeddings on large graphs [167]. This inductive capability is achieved by sampling a fixed-size neighborhood of nodes and applying an aggregation function over these neighbors' features. Several aggregation functions, including mean and pooling aggregators, can be used in GraphSAGE, allowing flexibility in capturing different levels of neighborhood information. The embedding for a node v is computed by aggregating the features of its neighboring nodes and combining them with the node's features. Formally, the GraphSAGE update rule for a node v at layer k is given by the equation 2.1.

$$h_v^{(k)} = \sigma\left(W^{(k)} \cdot \text{AGGREGATE}\left(\{h_u^{(k-1)}, \forall u \in \mathcal{N}(v)\}\right) + W_{\text{self}}^{(k)} \cdot h_v^{(k-1)}\right) \tag{2.1}$$

$\mathcal{N}(v)$ denotes the neighbors of node v, $h_v^{(k)}$ is the embedding of node v at layer k, $W^{(k)}$ and $W_{\text{self}}^{(k)}$ are trainable weight matrices, and σ is a non-linear activation function as usual for NNs.

Because models are suitable to be represented by graphs, our contribution (cf. Chapter 5) presents a potential way to apply the use of HGNNs for MDE.

2.2.3 Large Language Models

LLMs are NNs created with the transformer architecture [154], pre-trained on massive textual content corpora coming from the internet and other sources. They are tailored for text completion, generation, and natural language understanding. Together with other *foundation models*, they are studied as part of what is called *Generative AI* [185]. Essentially, given textual inputs, i. e. the *prompts*, they generate corresponding text outputs probabilistically, producing high-quality text that often mimics human-like writing.

Base LLM models are initially trained to predict the next word based on large text datasets, using context to generate the most likely subsequent token. For example, a simple prompt like "What is the capital of France" may result in predictions related to French cities, population statistics, or trivia, as the model's output reflects its general training data. In contrast, an *instruction-tuned LLM* has undergone additional fine-tuning, specifically to follow user instructions [186]. This distinction significantly improves the model's ability to respond to questions. Using the same example, a tuned model is much more

likely to output the specific answer, "The capital of France is Paris." Instruction tuning involves supervised fine-tuning, where LLMs are trained on a corpus of inputs and expected outputs (instructions), followed by reinforcement learning steps such as Reinforcement Learning from Human Feedback (RLHF) [187], to refine and improve response quality further.

LLMs have seen rapid development over recent years, evolving from simple text-to-text language models like T5 [186] to architectures like GPT-3 [160], which are capable of not only generating human-like text but also performing reasoning, translation, summarization, and many other complex natural language processing tasks.

Research continues to push the boundaries of LLM capabilities [185]. One area of exploration is how these models adapt to specialized tasks through instruction-tuning and transfer learning, as they can now handle domain-specific tasks like legal text generation or software code completion [188]. On the other hand, recent advancements have shown that as models scale in parameters and training data, they exhibit *emergent abilities*, capabilities that were not explicitly programmed or predicted [189]. Models like GPT-4, with hundreds of billions of parameters, [27] demonstrate advanced reasoning, arithmetic, and even coding skills without specific task training [169].

2.2.3.1 Prompt Engineering

Prompts are the primary means of interacting with LLMs. PE has emerged as a critical technique to maximize the performance of LLMs by systematically designing inputs that guide the model's outputs towards desired results [190]. Prompt design is crucial because it allows users to shape the model's responses without retraining them, improving efficiency and accessibility. PE involves empirical exploration, where the model's performance can vary across tasks, and it often relies on well-known heuristics and iteration to optimize results. For instance, incorporating clear task instructions, role-based directions, and specific formatting cues are common approaches in the ChatGPT [28] interface. [29]

Two key PE techniques highly relevant to our contributions are *Few-shot Learning* [191] and *Chain-of-Toughts (CoT) prompting* [192]. Few-shot learning enables the model to learn from a handful of high-quality demonstrations (examples of input-output pairs),

27. OpenAI does not disclose exact number and inner details
28. ChatGPT is the popular user interface provided by OpenAI to access their flagship LLMs through a chat-like interaction
29. `https://platform.openai.com/docs/guides/prompt-engineering/` *(Last Accessed in November 2024)*

improving results compared to a Zero-shot approach where no examples are provided. CoT prompting, on the other hand, introduces a sequential reasoning process by structuring the prompt into a series of smaller steps, which helps the model break down complex tasks into manageable subcomponents, leading to more reliable and interpretable outcomes. Many other prompt techniques have been studied for different purposes [193]. Still, they are not directly used in the scope of this thesis, even posing potential improvements for future work (cf. Chapter 7).

2.2.3.2 Tool-augmentation through the use of LangChain

LLMs are increasingly integrated with external tools and systems to enhance their functionality and utility. Frameworks such as LangChain [194], [30] Llamaindex, [31] and DSpy [32] provide the necessary infrastructure to develop composable applications powered by LLMs. LangChain, in particular, offers an open-source approach to composing multiple LLM calls in a structured manner, effectively applying the composite design pattern [195]. This allows developers to link together different components, such as prompt templates, model outputs, and third-party tools, creating a rich ecosystem for building intelligent applications. The use of such tool-augmentation frameworks opens new opportunities for enabling LLMs to interact with dynamic datasets, APIs, and real-world systems [196, 197].

In conclusion, the rapid evolution of LLMs has positioned them as critical components in the broader landscape of AI, with profound implications across multiple industries and applications. Our contribution (cf. Chapter 4) presents a step further on their use for MDE.

2.3 Application of Model Views in the Industrial Use Case [33]

In this section, we present a collaboration between industrial and academic partners of the AIDOaRt project towards a model-based approach for CPS engineering at Volvo

30. `https://www.langchain.com/` *(Last Accessed in November 2024)*
31. `https://www.llamaindex.ai/` *(Last Accessed in November 2024)*
32. `https://dspy-docs.vercel.app/` *(Last Accessed in November 2024)*
33. Content partially published at J. Cederbladh et al., "Towards Automating Model-Based Systems Engineering in Industry - An Experience Report," in 2024 IEEE International Systems Conference (SysCon), Apr. 2024, pp. 1–8. doi:10.1109/SysCon61195.2024.10553610.

Construction Equipment (VCE). VCE is a leader in developing and producing solutions in the construction equipment domain. In particular, they are pursuing the electrification of their machines to move towards a more sustainable future. Relying on the experience of VCE engineers in terms of interoperability, adaptability, and automation of the design activities, we elicited a set of challenges and issues for a practical industrial use case.

To potentially handle these challenges, we propose an approach considering a combination of prescriptive modeling, model transformations, AI-augmented model views, modeling process mining, and AI-based modeling recommendations. This section first introduces the general use case and the identified challenges tied to the AIDOaRt goals. Then, we quickly explain the overall solution, mainly focusing on the EMF Views role and the potential use of its AI-augmented features. Details on the other performed implementations and overall evaluation can be checked on the original paper by Cederbladh *et al.* [66].

2.3.1 Introduction to the VCE use case

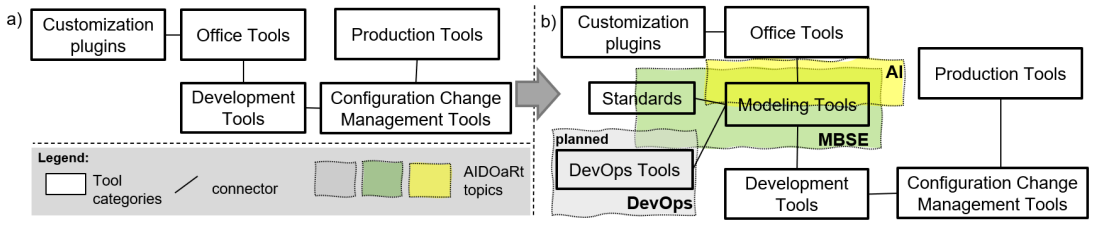

Figure 2.15 – Improving the automation of the VCE engineering process as envisioned in AIDOaRt.

With increased customer and regulatory emphasis on sustainability, VCE is on a transformation journey focusing on electrifying its construction machines, including battery-electric and fuel-cell technologies. Maintaining quality is paramount even during the transition period, with fast prototyping and short lead times. This requires the application of new technologies, not only in the final product but also during development. Through the AIDOaRt project collaboration, VCE provided an industrial use case in the form of a *Dumper System (DS).* [34]

Figure 2.15a shows a high-level tool landscape and their inter-dependencies (lines) as currently used in practice for the software and system engineering of VCE products

34. `https://www.volvoce.com/united-states/en-us/products/articulated-haulers/a60h/` *(Last Accessed in November 2024)*

and their variants, like the DS mentioned above. The engineering process starts with office tools, e. g., Visio and Excel by Microsoft, which are extended with useful plugins (e. g., architecture description palettes with variability aspects) to produce requirements and architectural definitions. Then, VCE experts manually inspect the resulting artifacts as guidance for coding software components and specifying simulation models of physical components of their products. Variants offered by VCE product lines are finally configured via change management tools.

Figure 2.15b shows the expected improvement from a tool landscape perspective thanks to the AIDOaRt project collaboration. New contributions brought by the AIDOaRt consortium (wavy boxes) are considered to foster the automation of the engineering process. Model-Based System Engineering (MBSE) [35] techniques and practices are explicitly introduced, with modeling tools pivotal in transforming descriptive engineering artifacts produced by office tools into models. The objective is to pave the way for integrating MBSE, AI/ML, and DevOps techniques and practices. Transforming current descriptive artifacts to a prescriptive model-based representation is a suitable step toward relieving and improving many bottlenecks in current processes.

2.3.2 Identified Challenges

We identify key challenges for a model-based approach to architecture descriptions from an industrial perspective.:

CH1: Managing interoperability and traceability in the system development process: CPS engineering is a multidisciplinary process and usually requires the integration of DSLs and tools. Thus, interoperability is a major concern, and it can be realized by weaving techniques applied to artifacts produced at each stage of the system lifecycle. In this respect, traceability is another key indicator that tests the quality of the produced artifacts throughout the development process, i. e. from the requirements gathering to the actual development. This challenge directly relates to the general motivation for our contributions (cf. section 1.2).

CH2: Promote the adoption of modeling practices in an industrial context: Adopting modeling practices within the industry is a common long-term challenge in the modeling community. Thus, it is naturally also reflected in this case study. To promote

35. This section uses MBSE instead of MDE in respect to the internal use by VCE and other AIDOaRt partners. Indeed, models at VCE do not necessarily drive the full process, and they use models for different purposes

modeling in the VCE context, the proposed solution architecture needs to demonstrate the added value of modeling in this case. Specifically related to the use of model view solutions, we expect that our answers to the **RQ3** (cf. section 1.2) can contribute to identifying how the use of DL can contribute to increasing its adoption.

CH3: Supporting automation through the combination of MBSE practices and AI-based tools: VCE engineers manually specify the system components through loosely integrated tools. Even though the engineering process is conducted correctly, MBSE practices can ease the burden of manual specification by offering a plethora of utilities. Furthermore, modeling activities can be automatized by employing AI-based algorithms. Therefore, adopting MBSE practices can improve the whole process by reducing the manual effort required by the VCE engineers. The correct application of AI techniques to improve MBSE practices and tools permeates both of our main contributions. Still, for this identified challenge, we highlight our approach to (partially) automate the view creation process using DL as described in Chapter 4 and directly related with our **RQ1** (cf. section 1.2).

CH4: Handling legacy artifacts: Legacy artifacts created by VCE engineers are valuable assets and play an essential role in specifying new CPSs, mainly since the current engineering methods, processes, and workflows rely on past expertise. On the one hand, a novel system must support the integration of legacy artifacts and the development of critical components. On the other hand, to be acceptable, new engineering practices must not disrupt well-established routines. Therefore, a flexible solution that integrates legacy and new approaches and artifacts is needed. Precisely on the use of model views, the contribution described in Chapter 5 directly relates to using legacy artifacts for training an DL algorithm. It is directly used to help in the answer for **RQ2** (cf. section 1.2) and, even though it is not directly associated with the integration of this legacy information for the use of engineers, it is an efficient way on how to reason on the legacy data for new purposes.

To cope with these challenges, we proposed integrating solutions from several partners with external (open-source) Computer Aided Software Engineering (CASE) tools. The proposed approach notably aims at leveraging MBSE and AI/ML capabilities for i) capitalizing on legacy engineering data, ii) supporting the structural modeling of CPS architectures and its variants via the SysML™ [88] and AutomationML [198] standards, and iii) allowing modeling recommendations via ML and process mining techniques. Our initial findings show that our approach can help improve design operations' automation,

which is part of the typical VCE workflow. The following section presents an overview of the proposed approach, focusing on its use of model views.

2.3.3 Overview of the Approach

The use case concerns the need to provide *system modeling* capabilities to industrial practitioners playing the role of *domain experts*. Existing office tools, like Visio and Excel, provide *descriptive system modeling* capabilities. The current practice at VCE leverages Visio documents for *graphical representation* and Excel sheets for *variant descriptions*, both documenting product lines with components and their variants. Office tools' availability of industrial-grade APIs provides generic automation capabilities, resulting in complex implementations of customization plugins and poor automation results. To overcome this, the approach proposed by AIDOaRt partners to VCE aims to enable *prescriptive system modeling* capabilities.

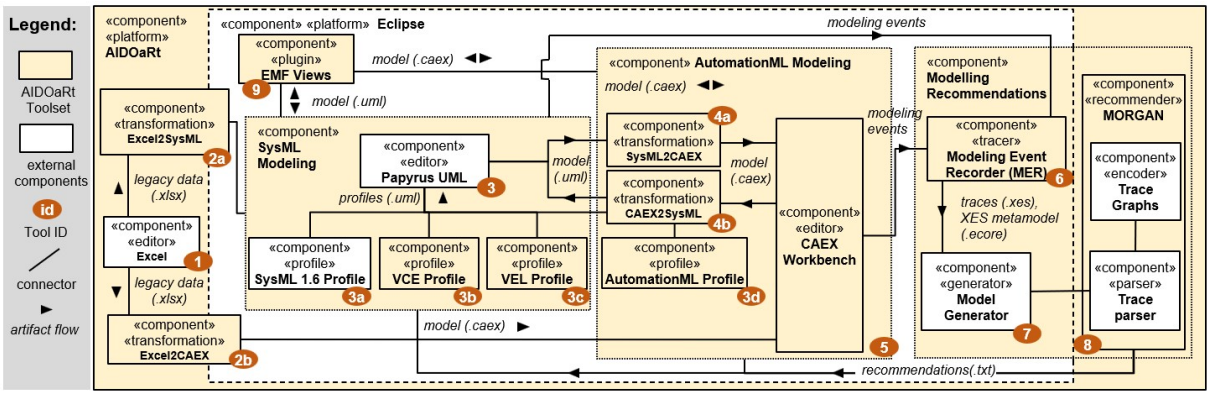

Figure 2.16 – The solution architecture for VCE challenges as part of the larger AIDOaRt framework

Figure 2.16 depicts a solution architecture detailing the generic support for the landscape sketched in Figure 2.15b. It integrates partners' solutions, as offered to the whole AIDOaRt consortium, with open-source tools and newly developed components, like model transformations.

In the presented use case, EMF Views supports *Multi-view Modeling*, conceived as an engineering activity independent from specific modeling languages. This way, it can be applied to federate any EMF-based models involved in the CPS engineering process. Currently, it is used to federate SysML™ and AutomationML (AML) models into integrated views that VCE engineers can navigate and query depending on their needs.

link.vpdl ×

```
1  create view simple_link as
2
3  select CAEX.CAEXFile.*,
4         CAEX.InstanceHierarchy.*,
5         CAEX.InternalElement.*,
6         SysMLBlocks.Block.base_Class,
7         UML.Class.*,
8         UML.Class join CAEX.InternalElement as problematicBlock
9
10 from 'http://www.dke.de/CAEX/300' as CAEX,
11      'http://www.eclipse.org/papyrus/sysml/1.6/SysML/Blocks'
12     as SysMLBlocks,
13     'http://www.eclipse.org/uml2/5.0.0/UML' as UML
14
15  where s.name = t.name for problematicBlock
```

Figure 2.17 – VPDL snippet for creating a simple view relating a SysML™ (UML-profiled) model and an AML (CAEX) model with EMF Views (Screenshot).

Figure 2.17 shows an example of such a view definition as expressed using the *VPDL*. In the present view, we want the elements of type *CAEXFile* to appear with all their properties. The SELECT part also declares the new inter-model associations to be added to the view (JOIN). In our example, we want to create a new association named *problematicBlock* between the SysML™ *Block/Class*es and the AML *InternalElement*s. Thanks to this, the VCE engineers can more easily get an overall vision of the system under study and make design decisions accordingly without referring to legacy data from Excel sheets.

2.3.4 Initial results

Using standard modeling languages paves the way for further connection to downstream activities. In particular, using system modeling languages such as SysML™ or AML enables domain experts to unify artifacts via shared models. Using EMF Views on the described context is a potential way to address specific challenges **CH1** and **CH2**. It improves interoperability via standardized means of representing data.

Initial results have been extracted from the AIDOaRt hackathons [36] and initial evaluation from VCE engineers [66] showing show promising results. However, AI-based techniques are more globally relevant for industrial practitioners when modeling their CPSs, which means that a AI-augmentation of the used tools can be helpful. Several components are already partially automated, and the contributions presented in this thesis go further in this direction, mainly within the challenges described in the following section.

36. Descriptions and details of the Hackatons were published on Linkedin`https://www.linkedin.com/newsletters/hackathon-challenges-6985181015648444416/` *(Last Accessed in November 2024)*

2.3.5 Challenges and Opportunities on the Use of Model Views

Beyond choosing modeling languages, engineering complex CPSs involves creating, transforming, and using multiple models describing various complementary system aspects [11]. As shown earlier, model views approaches provide unification mechanisms to federate and manipulate such heterogeneous models in a more transparent way [21]. Once built, model views can be used to uniformly navigate, query, and transform the aggregated data from the various contributing models. In the context of AIDOaRt, we propose to rely on *EMF Views* as a scalable and computationally efficient approach to create and handle model views [42].

The identified challenges (CH1–CH4) highlight the need for advanced approaches to address interoperability, promote modeling adoption, support automation, and manage legacy artifacts in the engineering of industrial CPS. These challenges align with the overall objectives of this thesis. We argue that integrating AI-augmented model views enables scalable and adaptive handling of multi-model interactions, addressing complexity challenges and enhancing efficiency in CPS engineering workflows.

2.4 Summary

The presented background chapter provides a basis for understanding the contribution chapters. We first introduced and briefly explained core concepts surrounding MDE, focusing on model views as a mechanism for handling the complexity of models by abstracting specific perspectives relevant to particular stakeholders or tasks. We also detailed the EMF Views tool as an implementation supporting model views within EMF alongside an industrial case study demonstrating practical challenges and motivating the need for AI-augmented model view solutions.

In the DL section, we focused on the basic concepts behind NNs and details on LLMs and GNNs, which will be used respectively on Chapters 4 and 5. We argue that the DL's ability to learn complex patterns enables it to automate parts of model view creation, potentially helping how views are defined and managed in engineering complex systems such as CPSs.

In the next chapter, we provide the state-of-the-art on both model views approaches aligned with our research and the primary research efforts and open challenges when applying DL techniques within MDE.

Chapter 3

State of the Art

This chapter complements our background presented in Chapter 2 with the state-of-the-art study on the two main topics discussed. Firstly, the section 3.1 presents a mapping study of notable model view solutions that can be interesting to put our contribution in context. It is complemented by the state-of-the-art research on the use of DL for MDE(also broadly MBE) tasks, highlighting the challenges and opportunities of its use.

3.1 Model View Approaches

Various approaches have been proposed within MDE to support the definition, management, and manipulation of model views. These approaches mainly differ in handling heterogeneity, non-intrusiveness, and the dynamic evolution of models. From the work of Bruneliere *et al.* and Cicchetti *et al.*, we can have an extensive overview of the current scenario of model view approaches [19, 21]. Based on their work and an updated search of the most recent developments in the area, we analyzed model view approaches that align with the contributions described in this thesis, putting them in context with the current scenario. We checked approaches that provide tool support, mainly within the EMF. We are especially interested in three aspects of each approach:

— Language aspects: We are interested in the viewtype and query languages that define the model views. While the first defines which elements are allowed to be included in a given viewpoint of the system, [1] the second is used to reason on the models and compute the elements presented in the final view. For the scope of the thesis, we search for explicit view definitions, mainly through the use of DSLs or annotations.

— Virtualization mechanism for computing the views: The Viewtype/View manifestation can be materialized, e. g. with the copy/duplication of base models' elements,

1. For simplicity, the relation viewtype/viewpoint can be seen similarly as the relation metamodel/-model [21]

or they can be virtual, relying on proxies to the existing (meta)model elements. While we pay attention to approaches with materialized views, we focus primarily on tools that provide virtualization mechanisms.

— Relationships definition: The definition of relationships between elements across views (i. e. inter-model relationships) is performed using constraint languages like Object Constraint Language (OCL), weaving models, trace models, or through model transformations. These correspondences act as contracts that must hold across views, enabling automated checks for consistency.

Our interest in these specific aspects is due to better aligning with our proposed contributions. In the following subsection, we explore existing solutions that align with these aspects, describing their capabilities and relevance to the challenges tackled in this thesis.

3.1.1 Existing solutions

OpenFlexo project was created by the company "Openflexo SCIC" aiming to make digital tools more accessible, and bridge the gap between various business areas and IT [61]. At its core, OpenFlexo utilizes the internal Federation Modeling Language (FML) to provide model federation, creating a virtual view gathering data from different data sources, including spreadsheets, SysML™ diagrams, and PDF documents [61], for example. These data sources are often managed by their respective tools, reflecting diverse organizational practices and converted to behave as homogeneous models in the OpenFlexo context. OpenFlexo employs technological adapters to establish communication (i. e. inter-model relations) among these models, and they offer synchronization mechanisms between the view and their base models. Openflexo has proven to be efficient, in combination with other tools, in engineering critical systems [199].

Eclipse Epsilon framework is a family of modeling languages and tools that work with EMF and other models. *Epsilon Decoration* [200] uses annotations to extend models with additional information that defines views, keeping them lightweight and manageable. It allows developers to apply overlays to base models, introducing annotations that make particular concerns visible without altering the underlying model. There is no explicit viewpoint definition (i. e. viewtype). The view is neither materialized nor virtual since it is not explicitly defined. Since it is built on the Epsilon framework, the provided query mechanisms for Epsilon models also apply to the views, which allows for weak inter-model relationships, even those that are not explicitly defined.

ModelJoin is used as part of the *Vitruvius* framework [2] for view-based (software) development. Vitruvius was developed by the Dependability of Software-intensive Systems research group (DSiS) at the Karlsruhe Institute of Technology (KIT). ModelJoin [201] leverages a human-readable textual DSL to define editable views across heterogeneous models (i. e. models that conform to different metamodels). It allows users to define correspondences (i. e. inter-model relationships) and projections over different metamodels, making it possible to work across diverse modeling languages without breaking the separation of concerns inherent in each language. Since the views are essentially models created using the defined DSL, their manifestation is materialized.

Viatra is a framework that incorporates the result of a long-running research project mainly supported by the *incQUERYLabs* [3] and various industrial and academic partners. Essentially, Viatra focuses on reactive, event-driven model transformations [62]. The heart of Viatra's reactive behavior lies in the Viatra 3 Event-driven Virtual Machine (EVM) [63]. View mechanisms are provided through the *Viatra Viewers component* [202]. [4] Viatra Viewers [62, 63, 203] is built over the EMF-IncQuery used to support model querying. By focusing on incremental query evaluation, VIATRA Viewers help create views that can efficiently react to changes in the base models. It is efficient for dynamic environments where model consistency is essential. It achieves this by combining internal languages (e. g. Java and Xtend) and defining partially evaluated model queries. To provide inter-model relations, Viatra uses a specific trace model to store this information in a model. The view manifestation is virtual, and the links between the models are stored using the dedicated trace model. Viatra technologies were successfully incorporated into different modeling tools and applied in various applications [204].

Sirius [205] [5] is a workbench tool that allows users to create graphical modeling solutions over EMF models. Sirius is developed by the companies Obeo [6] and Thales. [7] With Sirius, developers can explicitly define viewpoints over domain-specific models, providing customizable visual representations for different stakeholders. Views are defined using a View Specification Model (VSM) that allows the definition through diagrams, tables, or trees. It can be complemented using Acceleo Query Language (AQL) for complex query

2. `https://github.com/vitruv-tools/` *(Last Accessed in November 2024)*
3. `https://incquery.io/` *(Last Accessed in November 2024)*
4. `https://eclipse.dev/viatra/documentation/addons.html#!#viewers` *(Last Accessed in November 2024)*
5. `https://eclipse.dev/sirius/` *(Last Accessed in November 2024)*
6. `https://www.obeosoft.com/en/` *(Last Accessed in November 2024)*
7. `https://www.thalesgroup.com/en` *(Last Accessed in November 2024)*

computations. The manifestation of the view is virtual.

EMF-Syncer is the main result of a research effort conducted at the University of Leicester [206]. EMF-Syncer [207] enables the creation of editable view models from program snapshots at runtime without materializing them. The focus is using MDE tools for legacy systems [206]. The tool uses both DSL and annotations to define the synchronization policy, which dictates how program snapshots are represented as views. Using a virtualization mechanism, EMF-Syncer employs lightweight proxies instead of duplicating model elements. The relationships between models' elements are established using synchronization policies and implicit feature mappings. Additionally, EMF-Syncer utilizes a "store of synced links" for incremental propagation, functioning similarly to a weaving model to maintain relationships during synchronization.

Heterogeneous Matching and Consistency management Suite (HMCS) [208] is a prototype for organizing heterogeneous models as a network of models through a virtual correspondence model. This process is created and automated through DSL. The view is materialized in the form of the correspondence model. The inter-model relationships are explicitly defined and automatically computed, depending on the use case.

Other important examples that similarly provide model views but are less aligned with our select criterion may include: Vitruvius platform [209], Triple Graph Grammars (TGG) [210], Orthographic Software Modelling (OSM) [211, 212], Architecture Analysis and Design Language (AADL) [211, 213] and blended modeling [214, 215]. In general, model views are one of the potential ways to achieve model federation [36, 41], i. e. linking heterogeneous models developed by different stakeholders providing integration mechanisms. As briefly explained in the introduction (cf. 1.1), it is worth noting that it is not the only way to do so, with other potential approaches [28, 43–54, 216], altough they fall outside the thesis scope.

3.1.2 Comparison with EMF Views

Compared to the presented approaches, we can highlight the following characteristics of EMF Views (cf. section 2.1.3):

— Synchronization from model to views: EMF Views ensures synchronization between views and base models by sharing the same instances through proxies. Modification propagation back to the original models exists, even when limited.

— Scalability: EMF Views' virtualization avoids duplicating elements, resulting in better scalability, especially when handling large models [42].

— Flexibility: EMF Views utilizes the same model virtualization mechanism for both viewpoints and views through weaving models.

The systematic literature review of multi-view modeling approaches by Cicchetti *et al.* highlights EMF Views as one of the most prominent approaches [19]. They note that EMF Views offers a unique perspective by using the same model virtualization framework for both viewpoints and views, distinguishing it from other approaches.

EMF Views have significantly contributed to model view approaches in the current MDE scenario. The versatility of EMF Views in both academic and industrial settings [42, 64, 65] facilitates our adoption of it to the specific needs of our research. While EMF Views offer significant advantages, the application of model view approaches still faces challenges, particularly when addressing the complexities of defining and maintaining views for dynamic and evolving systems. The following section explores these challenges in greater depth, opening the way for our contributions.

3.1.3 Challenges in Model View Solutions

Despite its benefits, model view solutions present challenges to be addressed by the MDE community. Both Bruneliere *et al.* and Cicchetti *et al.* [19, 21] reunited a sound number of challenges to be addressed, from which we derive two challenges aligned with the thesis objectives and research questions:

— Limited Automation Support: The scarcity of tool support per each view approach and the subsequent shortage of automation is a highlighted problem [19]. Lack of expressiveness of view definitions and the necessary learning curve to learn how to define them can hinder adoption in industry-scale systems.

— Systematic definition of inter-model relations: The links between models often demand manual definition, i. e. the user should use some special kind of construct (e. g. matching-rules) to establish them. There is no proposition towards the automatic inference of these links. This automatic inference can be a step further in dealing with view maintenance problems pointed out in [21].

To the best of our knowledge, different DL techniques have been used to deal with MDE problems (cf. section 3.2), but not directly related to model views, neither to multi-view modeling problems in general. We intend to show a potential path towards this direction with our contributions.

3.2 Deep Learning for MDE

To contextualize our contributions, we discuss the related work and state-of-the-art on the applications of both GNNs and LLMs applied to MDE. We start briefly discussing some challenges on the MDE, with potential AI automated solutions. Altough we consider the use of traditional AI for context, the main focus is namely on the use of ML and DL.

3.2.1 Traditional AI Techniques for MDE

Traditional AI techniques have been applied in MDE to automate relation discovery and optimize model transformations [217]. These approaches often rely on heuristic search and evolutionary algorithms to efficiently explore possible model configurations. For instance, Burdusel *et al.* propose a method for automatically generating search operators that maintain model consistency during search-based model engineering, eliminating the need for meta-learning or expert knowledge [218]. Similarly, John *et al.* introduce an optimization framework based on evolutionary algorithms, focusing on mutation operator properties [219].

In the context of model transformations, Model Transformations by Example (MTBE) has been extensively studied as a user-friendly approach for deriving transformation rules from inter-model mappings [220, 221]. Traditional AI techniques have been leveraged for MTBE, including search-based methods [222] and genetic-programming [223], demonstrating their effectiveness in automating transformation rule generation.

While these approaches have achieved success in MDE, recent advancements in ML, particularly in DL, offer new opportunities for learning-based automation in model-driven processes.

3.2.2 Machine Learning for MDE

ML has widely supported different SE activities [68], recently including LLM-based agents [224]. Intelligent Modeling Assistants (IMAs) have recently attracted the interest of the MDE community aiming to support the crucial necessity for automation. AI/ML has already been identified as a relevant way of addressing several challenges in this direction [20]. We can cite some strategies to support modeling activities in a high-level analysis. Burgueño *et al.* [225] proposed an architecture based on NLP for the autocompletion of partial models. Given a set of textual documents related to the initial model,

relevant terms are extracted to train a contextual model using several NLP techniques.

NEMO [226] supports the completion of BPMN models by exploiting the LSTM strategy. The approach encodes the modeling operations using a sequence-to-sequence decoder to predict the next modeling operation.

Weyssow *et al.* present a learning-based approach that exploits RoBERTa, a pre-trained neural network, to suggest relevant modeling language constructs [81]. The latter are first encoded as structured trees, then the RoBERTa model predicts the missing elements and provides the modeler with insightful domain concepts.

Model consistency management is a fundamental issue in MDE and thus creates the need for relevant model repair techniques [227]. As they require a *smart* automation, such techniques are natural candidates for AI applications [72]. Different approaches propose the use of rule-based ML [73], decision trees [74], or Reinforcemnt Learning (RL) [69, 228], to find the best sequence of actions for repairing a given model and reaching a sufficient quality level. Groner *et al.* propose the use of different ML techniques which are not based on NN to predict the execution time of ATL transformations [229]. In the following, we explore in more depth some specific applications more aligned with the challenges addressed in this thesis.

3.2.3 Learning Constraints and Transformations

Dang and Cabot [230] describes the InferOCL tool dedicated to the automatic inference of OCL constraints for conceptual models and metamodels based on examples. Their objective is to increase the precision of domain descriptions.

On the MTBE, Burgueño *et al.* recently proposed a generic NN architecture to support the automated inference of model-to-model and model-to-text transformations [231]. Their objective is notably to limit potential implementation errors. Since transformations can implement views, we pay particular attention to these approaches.

3.2.4 Graph Neural Networks for MDE

López and Cuadrado [79] focus on achieving structural realism by utilizing a deep auto-regressive architecture combining a GNN and a RNN. Their evaluation demonstrates that they are superior to existing generators in terms of structural realism, consistency, diversity, and scalability in generating new models.

In parallel, Di Rocco *et al.* [80] develop and experiment with the MORGAN tool, a

GNN-based recommender system designed to assist modelers in specifying metamodels and models. The MORGAN tool is part of the use case described in subsection 2.3.

3.2.5 Large Language Models for MDE

Due to their capabilities, LLMs have been primarily applied in SE to code-related tasks [232, 233]. This notably includes code generation, repair, completion, debugging, and testing. Besides code, LLMs have also been applied to deal with SE processes [234, 235] for instance.

Closer to our context, there is a long history of solutions for dealing with text-to-SQL generation [236]. These notably include the use of ML techniques such as in the TaBERT Language Model (LM) pre-trained on (semi-)structured tables [237] for example.

A more recent solution investigates the use of LLMs and PE (precisely few-shot prompting) to explore the text-to-SQL capabilities of the GPT-family models [238]. Also quite recently, a specialized LLM (cf. Codex) has been used to generate OCL code Abukhalaf *et al.*, and a general purpose LLM (GPT-4) to perform the same task [240]. The LLMs could directly generate relevant code in both cases. Although the precise information of the datasets used for trained closed-source models like the GPT ones does not exist, a cursory search on GitHub can reveal the amount of public code available [8] To use the EMF Views VPDL as an example, the existing public code base (for both OCL and SQL) is very significantly larger. As a result, we cannot expect an off-the-shelf LLM to generate VPDL code as it can already generate SQL or OCL (for example).

On the model views side, we can see the proposed solution for partially automated view creation based on existing source code [206], but without using any explicit ML. Similarly, the VIATRA framework also allows the creation of view-like artifacts without providing an explicit textual definition [62]. More recently, López *et al.* presented the `Text2VQL` that allows the definition of VIATRA Query Language (VQL) queries through natural language specifications.

3.2.6 Engineering of LLM-powered applications

Designing systems that integrate LLMs involve careful consideration of various architectural components since the pure use of an LLM may not be enough to deal with

8. ∼6 k OCL files and ∼163 k SQL files. Searched on October 2024 with the query: `https://github.com/search?q=path%3A.LANGUAGE_EXTENSION+context&type=code`.

complex tasks [242]. According to the taxonomy proposed by Händler, besides the agentic approaches, a goal-driven development for LLM-based systems can include task decomposition and orchestration [243]. Given the importance of PE for LLM-based systems, the prompt evaluation is an essential step in designing this kind of application, which leads to the development of different evaluation strategies and tooling for PE, e. g. the tool proposed by Arawjo *et al.* [244]. Tool augmentation appears as an essential role in the design of LLM-powered systems in recent development [196, 197, 245], although we identified no patterns for its design and use, due to the freshness of this study area.

The use of LangChain as the framework of choice for LLM-based systems and the orchestration of its components can be found in a wide range of applications that go from health-care [246] to the development of educational resources [247], being especially good to the development of Chatbot applications [248]. Besides this variety of applications, LangChain maintainers also maintain an updated webpage where they implement a variety of LLM approaches (including PE strategies and also including other components) from different papers (including pre-prints) using the framework, which contributes to its versatility in dealing with different applications. [9]

To the best of our knowledge, no research directly applies LangChain (or even its conceptual ideas) to solving model-view problems.

3.3 Summary

This chapter showed multiple approaches to implementing model views, underscoring the current challenges and demand for automated solutions that allow engineers to adopt and use them effectively.

The chapter concludes with a mapping study that reviews current applications of DL (mainly GNNs and LLMs) in MDE, putting our research efforts in context. Altough DL (and broadly ML) has been used to solve different MDE challenges, the application of both LLMs and GNNs to the particular case of model views seems to remain an underexplored area of research.

Given this scenario, the following chapters will help us discuss the answers to our research questions previously presented in section 1.2. Chapter 4 will introduce a LLM-based application that shows the feasibility of using LLMs to help with model views

9. `https://python.langchain.com/docs/additional_resources/arxiv_references/` *(Last Accessed in November 2024).*

in the metamodel level (i. e. the viewpoint), partially automating its definition. This is directly related to the **RQ1** (cf. section 1.2) and also led us a step further in overcoming the challenge **CH3** identified in our motivation use case on AIDOaRt (cf. section 2.3). Chapter 5 looks to the potential automation of the view computation on model level (i. e. the view) through the use of GNNs. This will help us in answering the **RQ2** (cf. section 1.2) and a potential approach to use legacy artifacts as identified in AIDOaRt context (cf. **CH4** on section 2.3). The discussions on both contribution chapters, together with the overall view on Chapter 6, intend to give us the necessary clues in answering the **RQ3**.

Part II

Contributions

Chapter 4

LLM-powered application to aid viewpoint description

4.1 Introduction

Previous Chapters already discussed a pain point in developing complex systems: the separation of concerns and the fragmentation of information among stakeholders [28, 29]. As discussed, model view solutions are suitable for combining and navigating such heterogeneous models more transparently [21].

There are several more or less automated ways of creating model views (cf. section 3.1). They often rely on the model view definition via a DSL and/or query language. However, when manually writing these model view definitions, it can be challenging to identify the language elements to be selected, associated, or queried. This is notably true when the concerned modeling languages are large or semantically distant. Thus, automatically generating model view definitions is challenging since it requires a certain level of understanding and reasoning on the input metamodels (i. e. modeling languages).

As a potential solution, different ML approaches have already been proposed to improve the support for model management operations [81, 228, 249, 250]. In particular, LLMs, such as BERT [159] and GPT-3 [160], have demonstrated their capability in code generation [251, 252]. In the MDE community, LLMs have also been used for automating complex modeling tasks [84] and providing recommendations [83].

In this Chapter, we detailed an in-context LLM-based approach to assist engineers in writing model-view definitions. In particular, we automatically generate drafts of model-view definitions by providing as input only minimal information on the modeling languages to be combined. We want to achieve this by using off-the-shelf LLMs: we do not want to perform any costly additional training on the LLM, even if the LLM has not been trained initially on the model-view definition language. Thus, we query the LLM with punctual questions about the structure of the view, and we combine the LLM answers to generate

the model view definition programmatically. The approach is completely in-context, i. e. it relies exclusively on PE techniques to improve reasoning capabilities [253], composability, and to enable tool-augmentation [254].

We developed a first implementation of our approach as an LLM-powered application enhancing the capabilities of the EMF Views model-view solution [64]. To this end, we leveraged the LangChain open-source framework for developing applications powered by LLMs [255]. We validated our approach by applying it to a selected set of model views. These model views, coming from the literature and open resources, are initially specified in the VPDL of EMF Views or as ATL model-to-model transformations. We evaluate the relevance of the generated model view definitions by comparing them with the original ones developed by humans. The results we obtained already show the feasibility and applicability of our approach.

This Chapter [1] is structured as follows. Section 4.2 motivates our work via a running example. Then, section 4.3 presents the proposed approach, while section 4.4 describes its current implementation. Section 4.5 explains the experiments we performed and the results of our assessment. Finally, section 4.7 concludes the chapter.

4.2 Running Example

This section presents our running example for the rest of the Chapter 4, a simple model view called *Book-Publication*. It comes from the EMF Views user guide [2] where it is used to explain EMF Views and VPDL. We selected this running example because its contributing metamodels are very simple, but the view definition contains not-so-trivial associations.

Figure 4.1 shows the `book` and `publication` metamodels, in graphical and textual format (in PlantUML [3]). `Books` have `titles` and `authornames` and contain `Chapters` that have their own `title` and `nbPages`. `Publications` are more general than books, and contain a `title`, an `author`, a `publisher` and a publication `year`.

1. Content partially published at J. W. Pontes Miranda, H. Bruneliere, M. Tisi, and G. Sunyé, "Towards an In-Context LLM-Based Approach for Automating the Definition of Model Views," in Proceedings of the 17th ACM SIGPLAN Int. Conf. Software Language Eng. (SLE'24), Pasadena, CA, USA, Oct. 2024, pp. 1-14.doi:10.1145/3687997.3695650.
2. `https://www.atlanmod.org/emfviews/manual/user.html` *(Last Accessed in November 2024)*
3. `https://plantuml.com/` *(Last Accessed in November 2024)*

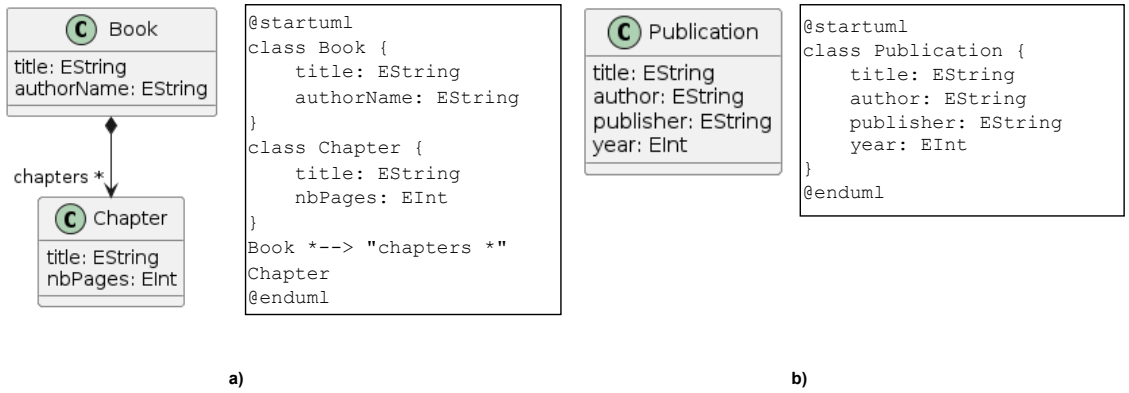

Figure 4.1 – Running example's metamodels in graphical and PlantUML format: a) book and b) publication

```
1  create view publicationsAndBooks as
2  select publication.Publication.*,
3      book.Book.*,
4      book.Chapter.title,
5      publication.Publication join book.Chapter as firstChapter,
6      publication.Publication join book.Chapter as bookChapters
7  from 'http://publication' as publication,
8      'http://book' as book
9  where s.title = t.eContainer().title and
10     t = t.eContainer().chapters.first() for firstChapter,
11     s.title = t.eContainer().title for bookChapters
```

Listing 4.1 – Example of a standard VPDL file

Listing 4.1 shows our view expressed in the VPDL language. The `select` part is used to define which concepts and properties from the `book` and `publication` metamodels have to appear in the view, i. e. `Publications` and `Books` with all their properties (`*`), `Chapters` with only their `title`. It also introduces new inter-model relations, i. e. the `firstChapter` and `bookChapters` relations between the `Publication` concept from the `publication` metamodel and the `Chapter` concept from the `book` metamodels. The `from` part allows users to declare the contributing metamodels, i. e. `book` and `publication`. Finally, the `where` part contains OCL-like expressions specifying matching rules for the new inter-model relations, i. e. for `firstChapter` and `bookChapters`. Using `firstChapter`, an OCL rule checks the inclusion of the first chapter of a book in the corresponding publication.

Note that the variables `s` and `t` denote the source and target element of the association (respectively).

To summarize, a VPDL file includes the three main parts of a model-view definition:

— A list of concepts and properties to be selected from the contributing metamodels (in the `select` part).

— A list of new relations to be created between the contributing metamodels (also in the `select` part).

— Queries describing how to compute these new relations on contributing models (in the `where` part).

Our objective is to be able to automatically generate such a model view definition by providing only limited input information. To this end, Listing 4.2 shows a possible user prompt we could consider for our running example.

```
"""
These two metamodels represent collections of books, but they have a different level of detail.
While the book metamodel details each chapter, the publication has more information about the publisher and publishing date.
I want to select all the information regarding publications, but I also wish to access the book chapters through the publication.
"""
```

Listing 4.2 – Possible simple user prompt for generating a corresponding model view specification.

As a result, Listing 4.3 shows a model view definition draft, which can be generated automatically from the previous prompt using our framework.

```
create view PublicationAndBook as
select Book.Book.*,
Book.Chapter.*,
Publication.Publication.*,
Book.Book join Publication.Publication as BookToPublication,
Book.Chapter join Publication.Publication as ChapterToPublication,

from 'http://book' as Book,
    http://publication' as Publication

where \`Combine Book and Publication based on the title and authorName/author
```

```
    ↪ attributes. This allows linking books to their corresponding publications
    ↪ .\‘
12      for BookToPublication
13 \‘Combine Chapter and Publication by associating chapters with their parent book
    ↪ ’s publication. This allows accessing chapters through the publication of
    ↪  the book they belong to.\‘
14      for ChapterToPublication
```

Listing 4.3 – Possible VPDL file resulting from the generation made with the prompt in Listing 4.2.

As visible in Listing 4.3, the approach aims at generating a draft of the `create`, `select`, and `from` parts of the model view definition. Moreover, it gives a natural language description of the content of the expressions to develop in the `where` part. We want to generate a syntactically correct draft of the model view definition. We also want to guarantee correct references to classes and properties of the original metamodels.

While the generated draft may contain semantic inconsistencies, it provides a valid, practical starting point for the engineer who will write the model-view definition. It partially relieves the developer from identifying which classes and properties of the original metamodels should be included in the view and which classes should be connected by inter-model associations. The queries (`where` part) are left to the developers. Still, the provided textual guidance can help them be more efficient when writing queries.

4.3 Approach

4.3.1 A Note on Fine-tuning and RAG

The performance of off-the-shelf LLM on a given task strongly depends on how much the task is covered by their training dataset [256]. To extend the application of LLM to tasks that require additional task-specific knowledge, the two most common techniques are fine-tuning and Retrieval-Augmented Generation (RAG).

Fine-tuning enhances an LLM, already pre-trained on a vast and diverse corpus of text, by additional training on new task-specific content. It refines the LLM model with specialized datasets relevant to the targeted task [186]. RAG enhances the standard LLM response for specific contextual data. It allows the injection of such data for the targeted task by indexing it in a vector database and making it directly accessible by the LLM [257].

Both techniques show promising results. Still, fine-tuning demands a large dataset of examples and high computational resources [257]. While more accessible, RAG applications still need a reasonably large dataset and an infrastructure for the retrieval process [258]. The availability of public datasets is a well-known problem in MDE, mainly when related withML tasks [259]. A few examples are publicly available, especially for view definition. Thus, in the presented approach, we do not use any of these techniques, and we study a solution that works directly on off-the-shelf LLM. Users only provide minimal information as input, e. g., the metamodels contributing to the view, to automatically obtain a draft of a corresponding model view definition (cf. section 4.2).

4.3.2 Overview of the Proposed Approach

Figure 4.2 provides an overview of our proposed approach.

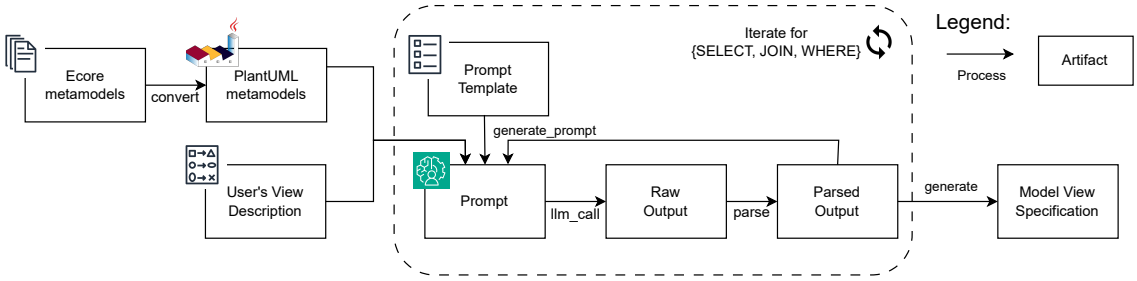

Figure 4.2 – Overview of the proposed approach

In EMF Views, the contributing metamodels are serialized in Ecore using the XML Metadata Interchange (XMI) format. However, their training makes off-the-shelf LLM more efficient for human-readable textual formats. Thus, we decided to use *PlantUML class diagrams as the representation format for metamodels*. It is a popular format supposedly included in LLM training sets and whose usage for LLM was already experimented in [84]. To this end, the first step converts *Ecore Metamodels* into equivalent PlantUML class diagrams. As an illustration, Figure 4.1 shows the two contributing metamodels of our running example (i. e. `book` and `publication`) converted to PlantUML.

Then, the *PlantUML metamodels* are injected with the user-provided prompt-like *View Description* into a specific *PE* created to solve a particular part of the decomposed problem. For our model view definition problem, we considered three complementary sub-problems. These sub-problems directly correspond to the main parts of the definition as introduced in section 4.2: `SELECT`, `JOIN` (associate), and `WHERE` (query). As a consequence,

in our case, we need to have three different well-crafted *PE* resulting in the LLM calling chain to be executed three times.

The performed actions are similar for each PE. First, a *Prompt* is generated from the concerned *Prompt Template*. The LLM is then directly called with this *Prompt* provided as input. As a result, it produces a corresponding textual *Raw Ouptut*. This *Raw Ouptut* is parsed to validate it and produce the textual *Parsed Output* in the expected format. This *Parsed Output* can be stored and, when required, reused as a complementary input to another iteration of the whole chain of actions. In our context, the validation process carried out by the parse operation is performed by specialized tools that deal with EMF models (e. g., PyEcore or the Java EMF API). If the output is not validated, the LLM is asked for a new solution. This is an example of tool augmentation to enhance the output quality.

Finally, the *Parsed Outputs* resulting from the different iterations (three in our case) are combined to generate the content of the target *Model Views Specification* (definition) textual file in the VPDL language in our case. For this final step, we do not use the LLM (since we do not assume it to be familiar with the VPDL syntax) but a standard code generator. In any case, the final resulting file is meant to be manually checked and eventually revised by the user before being provided as input to the Model View solution.

4.3.3 Focus on Prompt Templates

As presented earlier, *Prompt Templates* are critical artifacts of the proposed approach and related process. Listing 4.4 shows one of these templates, corresponding to our approach's JOIN (associate) iteration. This example notably illustrates how such templates are structured and what information they contain.

```
"""
You are now a PlantUML analyst that find relations between classes
 from two metamodels.

# TASK
Your task is to analyze the input metamodel and the view
description and define a list of relations between the metamodels'
 classes.
The classes are always combined in pairs, being one coming from
the first metamodel and the other coming from the second metamodel
.
```

```
Classes can be combined when they represent the same domain object or when they are complementary classes, which means that one can be extended with the attributes of the other.

Other possible reason for combinations is when the view description includes explicit attribbutes from one metamodel that should appear in the other.

Your answer should be a valid JSON list of dictionaries where each dictionary entry represents a relation.
It should be a list even when it contains just one relation.
Each relation always contains precisely one class coming from each metamodel.
In your response, the classes are always in order: the first class comes from the first metamodel, and the second class comes from the second metamodel.

# OUTPUT DATA FORMAT
{format_instructions}

# RULES
When generating the JSON response, you should follow these rules:
- Only use class names that exist in the metamodels. Never include classes that are not in the metamodels
- The relation's name can be any string, but it should be unique and meaningful for each relation.

# STEP BY STEP PROCESS
1. Identify all the classes from the first metamodel.
2. Identify all the classes from the second metamodel.
3. Given the metamodels and their classes, combine the elements in pairs when the selected classes represent the same domain object in each metamodel.
4. Given the metamodels and their classes, combine the elements in pairs when some selected class in the second metamodel can be complemented by some chosen class on the first metamodel and vice-versa.
5. Analyse the view description to find out other potential relations.
6. Ensure that the classes are combined in pairs, one from each metamodel.
```

```
7. Ensure that the relation's name is unique and meaningful.
8. Ensure that all the classes exist in the PlantUML metamodels.
9. Create the JSON array with the combination pairs.
10. Provide the answer.

# EXAMPLE
Given the following metamodels and view description:
View description: "The view  should conatins the name, and email from the Customer and also the name of the item bought by they."
Metamodel 1:
@startuml

    class Customer {{
        +int id
        +String name
        +String email
        +String deliveryAddress
    }}

    @enduml
Metamodel 2:
@startuml

    class Item {{
        +int id
        +String name
        +String category
    }}

    class Order {{
        +int orderId
        +String orderNumber
        +Date orderDate
        +Date creationDate
        +String currentOrderStatus
        +String customerName
    }}

    @enduml

The result Relations should be:
```

```
71 {{
72     "relations": [
73         {{
74             "name": "itemBoughtByCustomer",
75             "classes": [
76                 "Customer",
77                 "Item"
78             ]
79         }}
80     ]
81 }}
82
83
84 You can think step-by-step, but your final answer should contain
   only the valid JSON and nothing else. Exclude any explanation or
   delimiter from the final response.
85
86 # INPUT
87 View description: {view_description}
88 Metamodel 1: {meta_1}
89 Metamodel 2: {meta_2}
90 """
```

Listing 4.4 – Python f-string used as prompt template in the JOIN step

We implement the CoT approach to design the templates and corresponding prompts, using few-shot examples for the format instructions. As presented in Listing 4.4, our templates follow a structure that contains a role definition (line 2), the task definition (line 4), the task downstream explanation (lines 5 to 9), the desired output format (line 17, to be replaced at runtime by the explanations on the expected JSON-like format), and finally, the step-by-step execution of the task (directly implementing the CoT approach).

Building a relevant prompt template is an empirical process that involves several attempts based on try-and-error calls to the LLM until reaching the target results. However, it is possible to benefit from the PE best practices coming from academia [190], LLM provider guidelines, or recently developed frameworks like ReAct [260]. [4]

4. The Appendix A includes all other prompts used in our experiments.

4.4 Implementation

We created a prototype implementation to validate our proposed approach to produce model-view definitions as VPDL files for the EMF Views model-view solution. Globally, the current implementation relies on the combination of the LangChain framework for chaining our actions, with the PyEcore library [5] to handle EMF models, and the GPT LLM as made available through the OpenAI API. [6] The prototype source code is available on GitHub. [7]

Figure 4.3 shows an overview of the current technical implementation of our approach. To start this, the `LangChain` package displays the main composable components of the framework that are relevant in our case:

- `PromptTemplate`: A runnable component that dynamically manages prompt structures by incorporating multiple documents and variables to generate prompt content.
- `LLMModel`: A runnable component configuring an LLM's properties, including its temperature, that influences the creativity and variance of the outputs.
- `OutputParser`: A runnable component that handles the parsing and validation of model outputs, thus ensuring adherence to the required formats and supporting retry strategies for self-reflection.
- `Tool`: Any external piece of software.
- `Chain`: The interface for invoking a sequence of runnable components, each responsible for a specific task, and calling tools when necessary.

As described in section 4.3, we created a specific prompt template for each iteration of the process we follow in our approach. These are the primary inputs for creating LangChain `PromptTemplate` instances. These instances are then in charge of injecting the format instructions in JSON Schema (the default format used in LangChain), converting the Ecore metamodels into their PlantUML equivalents, and calling the LLM. Figure 4.4 shows a screenshot of the LangSmith tool, the companion tracing tool of LangChain used to capture all the occurrences of this chain. The full trace can be checked online. [8]

The `LLMPoweredViews` package displays the model view-specific components we de-

5. `https://github.com/pyecore` *(Last Accessed in November 2024)*
6. `https://platform.openai.com/docs/models` *(Last Accessed in November 2024)*
7. `https://github.com/NaoMod/Towards-an-In-context-LLM-based-Approach-for-Model-Views` *(Last Accessed in November 2024)*
8. `https://smith.langchain.com/public/716a3e84-d344-42e8-bf82-5b337a8b7d9b/r` *(Last Accessed in November 2024)*

veloped to refine and complement the LangChain ones:

- `VPDLGenerator`: The integration `Chain` component in charge of collecting the required inputs (i. e. the metamodel paths and *View description*) and chaining all the necessary `Runnable` and `Tool` components.
- `EcoreLoader`: A component for loading *Ecore metamodels* to be used in *Prompt Templates*.
- `EcoreOutputParser`: An `OutputParser` for checking that classes and attributes returned by the LLM *Raw Outputs* are present in the input *Ecore metamodels*. In practice, it parses these outputs and repeatedly calls the LLM again until the *Parsed Outputs* are not valid.
- `VPDLText`: A `Tool` component for generating the final target *Model View Specification*, as a VPDL textual file, from the *Parsed Outputs* produced by SELECT, JOIN, and WHERE iterations in our approach.

`EcoreLoader` and `EcoreOutputParser` use handlers from the `PyEcore` package. The objective is to enable the components from the `LLMPoweredViews` package (and coded in Python) to handle the input *Ecore metamodels* properly.

Regarding LLMs, we currently use GPT-4o in our implementation as we generally observed better performances than GPT-3.5. However, our approach and its implementation (thanks to LangChain) are flexible, allowing choosing between different OpenAI models and possibly other LLMs.

4.5 Evaluation

To evaluate our implementation, we defined a dedicated benchmark using actual model-view definitions from the literature (section 4.5.1) and open-source model-to-model transformations (section 4.5.2). Regarding LLM parameters, we opted for a default temperature of 0 to be as close as possible to a deterministic behavior (and thus results). In practice, the evaluation was performed using LangSmith, [9] a DevOps platform dedicated to the tracing and assessment of LLM-based applications built with LangChain.

To better assess our results, we compare them with the ones obtained via a baseline solution for LLM-based generation of model views. In this baseline solution, we used ChatGPT (the ready-to-use version of the OpenAI LLM encapsulated in a chat interface) to produce the views in well-known languages (Query/View/Transformation (QVT) for

9. `https://www.langchain.com/langsmith` *(Last Accessed in November 2024)*

views, ATL for transformations). To this end, we consider two simple prompts tailored for ChatGPT that include the same inputs used in our experiments (cf. Listing 4.5 and Listing 4.6 respectively).

```
"""
Given the view description and the following PlantUML metamodels,
please provide the definition of view written in QVT.

View description: {view_description}
Metamodel 1: {meta_1}
Metamodel 2: {meta_2}
"""
```

Listing 4.5 – Simple prompt for generating views in QVT

```
"""
Given the transformation description and the following PlantUML
metamodels, please give me the ATL code for the transformations.

Transformation description: {transformation_description}
Metamodel 1: {meta_1}
Metamodel 2: {meta_2}
"""
```

Listing 4.6 – Simple prompt for generating views as ATL transformations

Since ChatGPT is a general-purpose application trained and fine-tuned for human-machine interaction, our prompts included some conversational constructs (e. g. "please" and "give me") together with our minimal inputs. The detailed prompts and results of a simple query in the ChatGPT platform were stored in the same repository as our source code. This collection acts as an experiment journal containing the pair prompt/completion and a link to access the recorded chat. [10]

4.5.1 Reproducing Existing Model Views

Table 4.1 shows the four model views we considered in the first part of our evaluation. They have been selected for their heterogeneity in terms of contributing modeling

10. The links are maintained by OpenAI. It is impossible to ensure how long they will keep it.

languages (i. e. metamodels) and their varying levels of complexity in terms of mappings.

Table 4.1 – Evaluated Model Views in VPDL

ID	Model View Description	Metamodel 1	Metamodel 2	Source
V1	"The Book metamodel details each chapter, while the Publication has more information about the publisher and publishing date... [3 lines]"	Book	Publication	EMF Views manual
V2	"The considered view combines a Runtime Log model (that conforms to a simple trace metamodel), a Source Code model... [4 lines]"	contentfwk	ReqIF	Example view in [64]
V3	"The views allow us to follow the evolution of an engineering system. It shows different versions of the same system modeled... [3 lines]"	caex	ecoreXES	Example view in [66]
V4	"The view aggregates all the models seen so far. This allows the system engineer to transparently point to the relevant information (spread in different models)... [11 lines]"	Traceability	B	Example coming with EMF Views [11]

For each model view, Table 4.1 displays an identifier (ID), a high-level description, the name of the two contributing metamodels, and its source (from literature or other sources). Note that we did not write the model-view textual descriptions. Instead, we directly extracted small explanations of the desired output from the source document (e. g. research article or documentation).

Concerning model-views *3* and *4* in particular, we slightly adapted the descriptions from the sources since these model-views initially concerned more than two contributing metamodels. We also restricted the evaluation of these views to two metamodels. Indeed, our current implementation supports only two contributing metamodels. This restriction is due to the context window size of the publicly available GPT-4o LLM, which prevented us from considering numerous large metamodels within a single model view.

Overall, for model views in VPDL, the performed experiments aimed at evaluating:

— How effectively our approach automatically identifies possible relations between classes from two contributing metamodels – `JOIN` (associate).

— How accurately our approach automatically identifies relevant attributes for each class of a given relation – `SELECT`.

— Which level of quality and understandability are exhibited by the automatically generated model view definition – `SELECT`, `JOIN` (associate), `WHERE` (query).

4.5.2 Inferring Semantic Equivalence

In a second experimentation, we focus on a specific but significant kind of view: the views concerning two existing models (conforming to different metamodels) and connections between *semantically equivalent* elements from these models. The *SELECT* part of such a view is trivial since we always select all classes and attributes of the corresponding metamodels. We aim to leverage the LLM to infer the *JOIN*, i. e. the identification of the semantically equivalent classes in the two contributing metamodels.

To this end, we consider a set of model-to-model transformations in ATL from existing work (cf. 2.1.1.3). We want to build a view that contains the full source model, the full target model, and the inter-model relation between corresponding elements in the two models (i. e., instances of source patterns and target patterns of the same rule application).

Table 4.2 shows the five model-to-model transformations we considered in our evaluation, in addition to the previously presented model views. They have been selected from the ATL Transformations Zoo [12] considering their diversity in terms of contributing modeling languages and the domains they cover. Again, the descriptions are textual snippets directly extracted from the transformation documentation.

4.5.3 Obtained Results

Table 4.3 and Table 4.4 show the quantitative results of our evaluation using a 1-shot prompt template for the model views in VPDL from Table 4.1. They display the detailed results for the predicted relations (`JOIN`) and the predicted attributes (`SELECT`), respectively. The **Our approach** columns indicate the means of three consecutive executions of the evaluation by using precisely the same inputs and configuration. The **ChatGPT** columns indicate the results of a single execution of our baseline solution for comparison purposes.

Similarly, Table 4.6 shows the corresponding results for the model views as ATL transformations from Table 4.2. However, as explained, this only concerns the case's predicted relations (`JOIN`).

12. `https://eclipse.dev/atl/atlTransformations/` *(Last Accessed in November 2024)*

Table 4.2 – Evaluated Model Views as Model-to-Model Transformations in ATL

ID	Transformation Description	Metamodel 1	Metamodel 2
T1	"The BibTeXML to DocBook example describes a transformation of a BibTeXML model to a DocBook-composed... [5 lines]"	BibTeX	DocBook
T2	"The Class to Relational example describes the simplified transformation of a class model to a relational database schema.[1 line]"	Class	Relational
T3	"The "Families to Persons" transformation describes a simple... [2 lines]"	Families	Persons
T4	"RSS is a format for syndicating news and the content of news-like sites. Atom is an XML-based file format intend... [4 lines]"	ATOM	RSS
T5	"This transformation presents a basic example where a tree is transformed into a list... [2 lines]"	List	Tree

Table 4.3 – Quantitative evaluation - VPDL matching relations between classes using 1-shot prompt templates

ID	Ref.	Our approach			Baseline solution (ChatGPT)		
		Precision	Recall	Syntactic Correctness	Precision	Recall	Syntactic Correctness
V1	2	0.50	0.50	100 %	0.00	0.00	0 %
V2	1	0.02	0.50	100 %	N/A	N/A	0 %
V3	1	0.00	0.00	100 %	0.16	1.00	0 %
V4	1	0.00	0.00	100 %	N/A	N/A	0 %

Table 4.4 – Quantitative evaluation - VPDL matching properties using 1-shot prompt templates

ID	Ref.	Our approach			Baseline solution (ChatGPT)		
		Precision	Recall	Syntactic Correctness	Precision	Recall	Syntactic Correctness
V1	8	0.58	0.58	100 %	0.66	1.00	0 %
V2	8	0.38	0.38	100 %	N/A	N/A	0 %
V3	52	0.07	0.07	100 %	0.00	0.00	0 %
V4	12	0.54	0.54	100 %	N/A	N/A	0 %

Table 4.5 – Qualitative evaluation

ID	**Match Rules**	**Human judge**	**LLM judge**
V1	Yes	Good	3
V2	Yes	Satisfactory	2.5
V3	No	Good	2
V4	No	Inadequate	2

Table 4.6 – Quantitative evaluation - ATL matching relations between classes using 1-shot prompt templates

ID	**Ref.**	**Our approach**			**Baseline solution (ChatGPT)**		
		Precision	**Recall**	**Syntactic Correct-ness**	**Precision**	**Recall**	**Syntactic Correct-ness**
T1	16	0.05	0.08	100 %	0.11	0.12	0 %
T2	6	0.30	0.38	100 %	0.33	0.33	0 %
T3	2	0.50	1.00	100 %	0.00	0.00	0 %
T4	3	0.00	0.00	100 %	0.5	0.33	0 %
T5	2	0.00	0.00	100 %	0.5	1.00	0 %

Table 4.7 – LLM as judge experiment

ID	**Human judge**	**LLM Judge**
T1	Satisfactory	3.00
T2	Satisfactory	3.00
T3	Satisfactory	2.33
T4	Satisfactory	2.67
T5	Satisfactory	2.00

Overall, *Quantitative Evaluation* concerns the fully automated evaluation performed thanks to standard algorithms provided by the LangChain ecosystem and corresponding customized functions:

— *Reference (Ref.)*: The number of considered relations between classes and selected properties in the VPDL case, and of considered relations between classes in the ATL case. This represents our expected results.

— *Precision*: A standard metric providing the ratio of relevant items retrieved/predicted/matched based on the total number of retrieved items. It measures the accuracy of the retrieved items.

— Recall: The ratio of relevant items retrieved based on the reference's total number of relevant items. It measures the completeness of the retrieval.

— Syntactic Correctness: The percentage of correct generated code from a syntactic point of view.

For the baseline solution, we considered a rough approximation since it was necessary to make some assumptions. By default, the code generated by ChatGPT was not VPDL or QVT code. Instead, it used a language hallucinated by ChatGPT. Some results are indicated as non-available (N/A) when the generated code was almost entirely irrelevant.

Qualitative Evaluation concerns the one-to-one comparison between the final outputs of our approach and the expected outputs (i. e. the reference model views). This is a manual evaluation of the overall quality of the obtained results by experts in the VPDL and ATL languages. Table 4.5 shows our qualitative analysis for the model views in VPDL from Table 4.1:

— *Matched Rules*: The overall quality (manually assessed) of the generated textual explanation for each identified relation. It can be *Good* (the engineer directly understands the semantics of the relation), *Satisfactory* (it requires her some effort), or *Inadequate* (it is very or too difficult for her) – `WHERE` (query).

— *Human Judge*: The overall quality (manually assessed) of the generated output, i. e. a VPDL file or a set of ATL relations. We use the same classification as from the previous metric – `SELECT`, `JOIN` (associate), `WHERE` (query).

— *LLM Judge*: The overall quality (LLM assessed) of the whole generated output, i. e. a VPDL file or a set of ATL relations. Using the same setup, the LLM receives an extra prompt to give a score from 1 to 10 concerning the generated output. 1 means that transforming the output into the reference demands considerable effort, and 10 means that this transformation is easy. Although not a standard practice yet,

the use of LLM-as-judge becomes more common [261] – `SELECT`, `JOIN` (associate), `WHERE` (query).

Table 4.7 shows similar metrics to assess the quality of the results for the model views expressed as ATL transformations from Table 4.2. In this ATL case, only the *Human Judge* and *LLM Judge* were considered. Indeed, it is not trivial to go from the list of predicted relations to the final ATL code used as reference.

4.5.4 Analysis of the Results

Concerning the *Quantitative Evaluation*, the *Precision* and *Recall* vary from 0.00 (no relevant prediction) to 0.58 (a decent number of relevant predictions) depending on the cases. Overall, our approach performs better when trying to predict selected properties (`SELECT`) than when trying to predict potential relations (`JOIN`). This seems logical since identifying semantic relations between concepts is a more challenging task. Still, for the relations (`JOIN`), our approach is currently more efficient in the ATL case than in the VPDL one. This could be explained by the fact that the LLM is, by default, more knowledgeable about the notion of model-to-model transformation (and ATL consequently) than about the idea of model views (and VPDL). This is coherent with our choice of using an in-context approach to avoid performing a pre-training phase (for both VPDL and ATL).

Our approach does not currently perform systematically better in terms of precision and recall than the baseline solution. However, it always succeeded in providing an output that was at least suitable in terms of syntax. This is already valuable from a user perspective, compared to the baseline solution, which was sometimes unable to provide actually exploitable code. Indeed, using standard ChatGPT requires the engineer to have solid PE expertise to improve the results and avoid hallucinations. Our approach's main objective is to completely hide this complexity from the regular engineer.

Concerning the *Qualitative Evaluation*, the *Matched Rules* and *Human Judge* metrics reveal that a majority of the obtained outputs are satisfactory from the perspective of the engineer. While still requiring human intervention, the generated drafts of model view definitions appear to be relevant starting points. The *LLM Judge* scores, that globally range from 2 to 3, provide a complementary perspective on the possibility of transforming relatively easily the obtained outputs into the reference code. This indicates that, while not always very close to the expected output, the desired model view definitions can be derived by considering a reasonable number of modifications.

To summarize, our results so far demonstrate that our approach already automatically generates exploitable model view definitions. In this sense, it appears relevant compared to a baseline solution relying on standard ChatGPT. While still improvable, the obtained results show the feasibility and applicability of the proposed approach and its current implementation.

4.6 Example of Improved User Prompt

The results of any LLM-based application are strongly related to the quality of its prompts. In our proposed approach, we ensure some quality of the main prompts (i. e. the Prompt Templates). In addition to the main prompts, user instructions can strongly affect the final result. For fairness purposes, the experiments presented in Section 4.5 considered only the most straightforward prompts copying views descriptions from its original papers/websites.

This section will show how the user prompt may affect the tool's final output. To do so, we again use the book/publication as the running example. Listing 4.7 shows an enhanced potential prompt to be used as input in our prototype. Different from the one used before, this one was crafted based on some trial and error attempts. Although we performed some experiments using another LLM (e. g. ChatGPT) and asking for improvements on the given user prompt, the best results were obtained through direct interaction with the tool.

```
"""
You're working with two metamodels: one for Publication, which
    contains general information like the publisher and release
    date, and one for Book, which provides details about books and
    their chapters.
Your goal is to combine this information into a single output so
    that you can access both publication details and specific
    chapter information together.
To do this, you'll retrieve all relevant fields from the
    Publication data and link them to the Book data by connecting
    the chapters to the publication based on matching titles.
You'll need to find both the first chapter of each book and all
    its chapters, allowing you to view both summary and detailed
    book information related to the publication.
"""
```

Listing 4.7 – Python f-string with the user prompt for Book/Publication that achieved better results

Tables 4.8 and 4.9 are structured similarly to the ones presented in our evaluation also using the same identifier. We can see that the improved prompt indeed affected our results positively, primarily related to the precision and recall of the selected attributes. Using the same enhanced user prompt in our baseline comparison with ChatGPT did not affect the result significantly. For completeness, the listing 4.8 presents the final VPDL draft generated by the complete execution of our prototype using the enhanced user prompt.

Table 4.8 – Quantitative evaluation - VPDL matching relations using improved user prompt

ID	Ref.	Our approach			Baseline solution (ChatGPT)		
		Precision	**Recall**	**Syntactic Correctness**	**Precision**	**Recall**	**Syntactic Correctness**
V1	2	0.50	0.50	100 %	0.00	0.00	0 %

Table 4.9 – Quantitative evaluation - VPDL matching properties using improved user prompt

ID	Ref.	Our approach			Baseline solution (ChatGPT)		
		Precision	**Recall**	**Syntactic Correctness**	**Precision**	**Recall**	**Syntactic Correctness**
V1	8	1.00	1.00	100 %	0.50	0.50	0 %

```
1    create view bookView as
2
3 select select Book.Book.*,
4              Book.Chapter.*,
5              Publication.Publication.*,
6              Book.Book join Publication.Publication as
                   ↪ BookPublicationTitleRelation,
7              Book.Chapter join Publication.Publication as
                   ↪ ChapterPublicationTitleRelation,,
```

```
8
9  from 'http://publication' as publication,
10     'http://book' as book,
11
12 where 'Combine Book and Publication by matching the Book's title with the
       ↪ Publication's title to link general publication information with book
       ↪ details.'
13       for BookPublicationTitleRelation
14 'Combine Chapter and Publication by matching the Chapter's title with the
       ↪ Publication's title to connect chapter details with the publication's
       ↪ general information.'
15       for ChapterPublicationTitleRelation
```

Listing 4.8 – Generated VPDL file resulting from the generation made with the prompt in Listing 4.7.

4.7 Conclusions

In this Chapter, we presented an in-context LLM-based approach intended to support engineers in writing their model view definitions by providing only limited information as input. The main objective is to prevent them from starting from scratch when dealing with such a task, independently from the modeling languages contributing to the view. To achieve this, we proposed adopting state-of-the-art PE techniques in our MDE context and combining them accordingly. The current implementation of our approach notably relies on the LangChain integration framework, GPT LLM, PyEcore library, and newly defined EMF Views-specific components. We validated the proposed approach and implementation by considering different model views, specified as VPDL files or ATL model-to-model transformations. The results we obtained already demonstrate the feasibility and applicability of our approach.

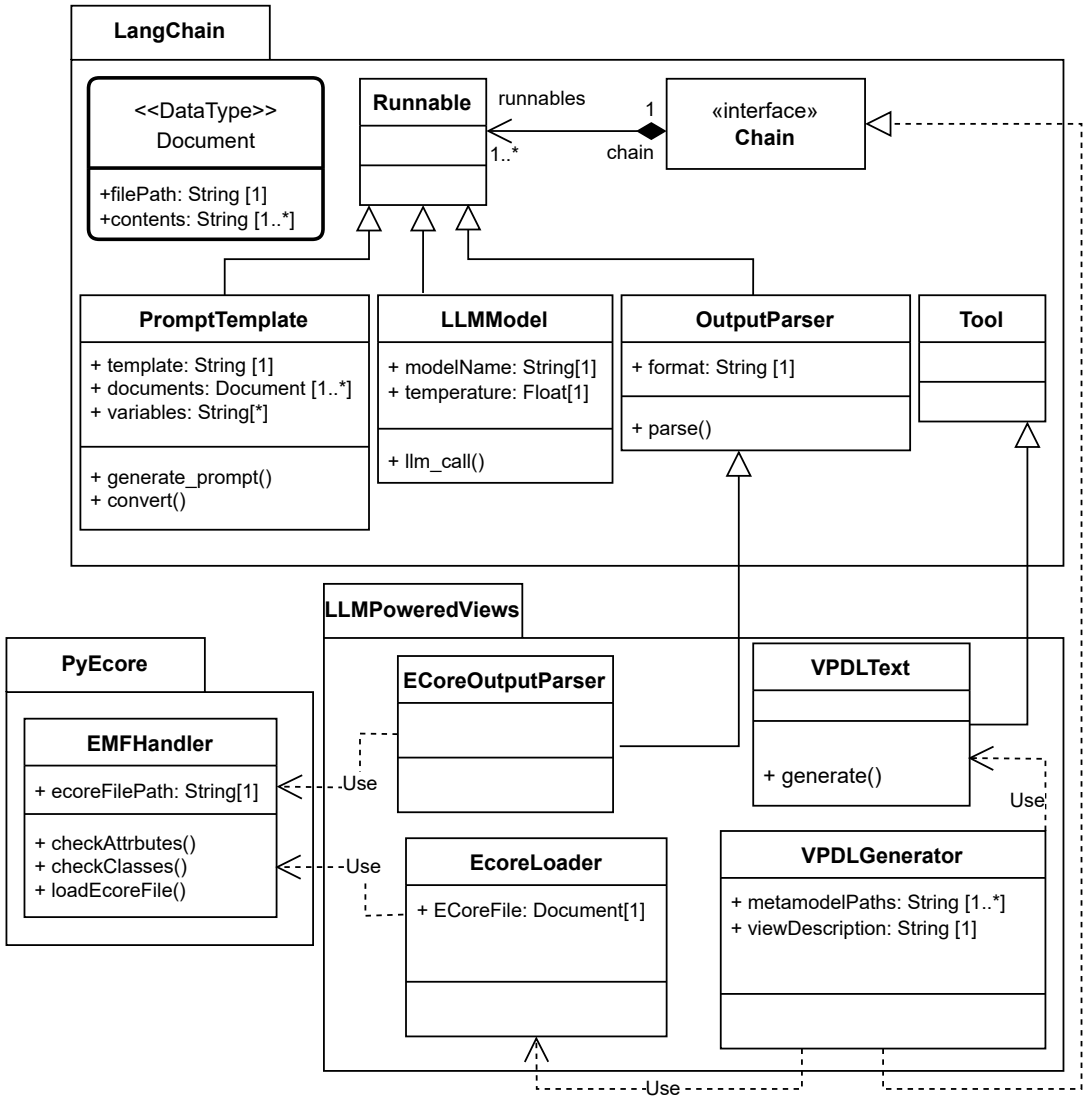

Figure 4.3 – Overview of the technical implementation of our proposed approach.

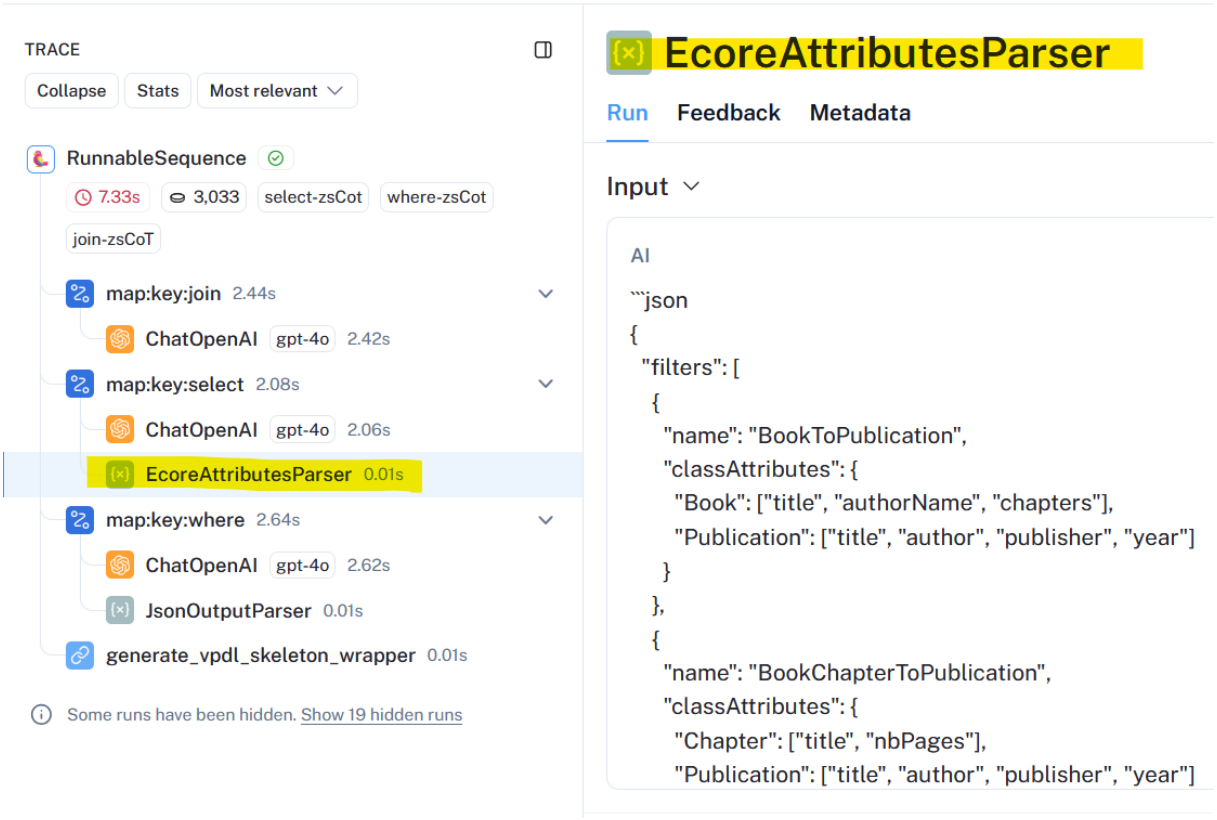

Figure 4.4 – Interface of the LangSmith tool in the context of our implementation.

Chapter 5

GNN-powered inter-model relations recommender

5.1 Introduction

In this Chapter, [1] we propose an DL-backed approach for computing model views that require the inference of inter-model links. In complement with what we have shown in Chapter 4 to help with the design phase of the view development, our objective is to simplify the view engineer's work as much as possible. It considers the demand of view computation during the runtime. To be coherent, we also want to hide implementation details so the view engineers do not need to write any ML code. To realize our approach during runtime, we rely on the link prediction capabilities of GNNs. Unlike previous research efforts (cf. section 3.2), we rely on HGNNs, the particular class of GNNs with native support for graphs whose nodes and edges have different types (cf. 2.2.2 for complete details). As a result, we extended the EMF Views [42] and its VPDL to integrate HGNNs. The engineer only needs to indicate 1) the relations to learning, 2) the parts of the models involved in the learning process, and 3) a relevant set of sample links. A declarative description of the architecture and configuration for the corresponding HGNNs is then automatically generated and can be manually updated. These HGNNs are transparently trained and finally used to infer inter-model links integrated into model views. We built a prototype of our approach and applied it to two sample case studies. We measured a promising accuracy in the inference of inter-model links.

The remainder of this Chapter is structured as follows. Section 5.2 introduces a running example. Then, section 5.3 presents the proposed approach, and section 5.4 describes its current implementation. Section 5.5 explains the evaluation we have performed. Finally, section 5.6 concludes the chapter summarizing our work.

1. Content partially published at J. Miranda, H. Bruneliere, M. Tisi, and G. Sunyé, "Integrating the Support for Machine Learning of Inter-Model Relations in Model Views," The Journal of Object

5.2 Running Example

This Section introduces the running example that will help us explain the proposed approach in detail. To be consistent, we first refresh the main concepts of link prediction using HGNNs (cf. Chapter 2 for complementary details). As initially established, our work focuses on the EMF Views model-view solution [42, 64] that uses the VPDL to specify viewpoints and build corresponding views (cf. Chapter 2 for details).

The systematic definition of how to relate concepts from different metamodels can be challenging. Even though we partially solved the problem of counting on LLMs to discover potential combinations, we still have the problem of defining the combination rules, mainly during runtime. Up to our current knowledge [19, 21], the related work focuses on the manual definition of rules for inter-model relations, i. e., matching rules (cf. section 3.1 for details). Sometimes, achieving these links with standard matching rules can be hard or even impossible. We intend to address scenarios where the engineers do not establish matching rules but infer inter-model relations from previous examples. To this end, we propose to rely on the link prediction capabilities of the HGNNs.

5.2.1 Link Prediction

As already mentioned in the background chapter (cf. section 2.2), (H)GNNs as inference machines can be used for various downstream tasks, e. g. node predictions, graph predictions, and link predictions. Our focus is on link prediction. Given a graph G, the link prediction task can be defined as computing the likelihood of observing a link in G between any two nodes v_x and v_y in V. Various techniques have been proposed to tackle link prediction, e. g., for similarity scores between nodes in social networks [262, 263] or recommendation systems [264]. Being those above v_x and v_y in V, the likelihood of a link between v_x and v_y can be given by a parameterized function $f_\theta(v_x, v_y)$. Considering that the embeddings u_{v_x} and u_{v_y}, f_θ can operate directly with these numerical vectors, so $f_\theta(v_x, v_y) \approx f_\theta(u_{v_x}, u_{v_y})$. In this scenario, the role of the HGNNs is to apply the message passing and aggregation layers on the embedding vectors to capture more complicated information based on node and edge types [265].

We argue that inferring inter-model links between contributing models of a view can be reduced to a link prediction task. In the following, we show that HGNNs can be used to learn a joint representation of the models and infer their links.

5.2.2 Example Details

We now introduce a running example to illustrate our proposed approach. We considered two simple but significantly large models with real-world inter-model links that we could use for training.

In our example, a Users model contains personal information on users that can be extracted from a social network. A Movies model contains information about movies that can be extracted from a film database. We want to automatically compute a model view containing users, movies, and links connecting each user with the *movies they probably watched*.

If we were building this view without any information on the movies actually watched by our users, we would embed some logical formula to estimate these links in the view. For instance, we could suppose that every user watched all movies tagged with the user's occupation (because we are all interested in movies that involve our job!). Of course such estimation formula may be arbitrarily complex (supposing that the viewpoint definition language contains a Turing complete expression language) and may take into account any information in the contributing models (i. e., the user profiles and movie database).

Let us suppose that we also have another large dataset, similarly structured, describing *other users*, but also including information about the movies each user watched. Now, we can try to automatically exploit this historical dataset better to estimate the inter-model links between our users and movies.

In practice, from the historical dataset, we want to automatically learn a mathematical relation between each user and the movies they watched. Then, we want this relation to be automatically applied in the view we are building to compute new inter-model links between our users and the movies they probably watched. Note that we are not interested in explaining the logic of the learned relation but only in obtaining as accurate inter-model links as possible for all users. We use HGNNs, transparently integrating them in the viewpoint definition language.

In our experimentation, we build our models using data from the well-known *MovieLens* dataset [266]. Given that the movie recommendation problem already has reasonable solutions using various techniques [267–269], our work is not about competing with these solutions. Indeed, later, we show that the estimation we obtained in this case study is accurate. However, our focus is not on obtaining a good estimation for links, as this is strongly dependent on the considered use case, the quality of the dataset, and the topology and parametrization of the HGNNs. Our contribution lies in integrating HGNNs in

the view definition language, aiming at increasing the usability of this technology by non-experts in ML. In particular, view engineers do not have to write any Python code, but only declarative specifications, to execute the training and inference of the HGNN.

Moreover, dealing with recommendations is only one possible application of our DL-backed model view approach.

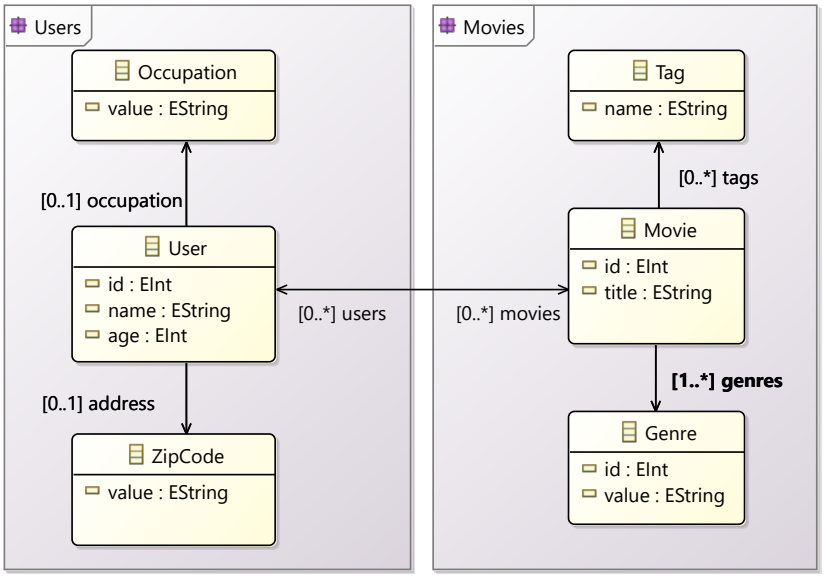

Figure 5.1 – Excerpts of the Users and Movies metamodels

Figure 5.1 shows excerpts of the two initial metamodels for this example (expressed in EMF Ecore). This metamodel represents `Users`, that are identified by an `id`, have a `name` and an `age`. Each `Movie` is identified by an `id`, has a `title`, and is also associated with a list of `Genres` and a list of `Tags`. A `User` may have watched several `Movies`, and a `Movie` may have been watched by several `Users`.

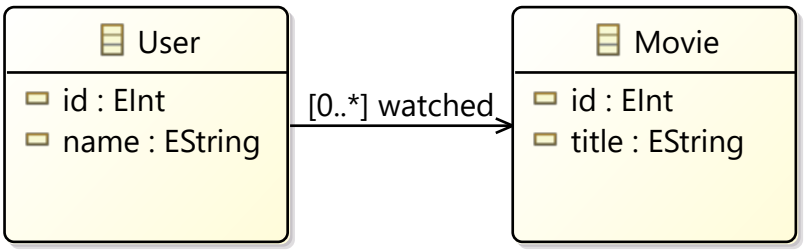

Figure 5.2 – UsersAndMovies viewpoint metamodel

Figure 5.2 shows the desired viewpoint metamodel. We want to obtain a view that

includes ids and names of Users, ids and titles of Movies, and a relation watched that lists for each user the movies they (probably) watched.

```
create view usersMovies as

select Users.User[id, name],
   Movies.Movie.*,
   Users.User join Movies.Movie
   as watched

 from 'http://paper/movies' as Movies,
      'http://paper/users' as Users

 where "t.tags->exists(tag | tag.name = s.occupation.value)" for watched
```

Listing 5.1 – VPDL file for defining a viewpoint for the running case using an OCL-like matching rule

Before our extension, VPDL allowed only to express logical formulas for estimating inter-model links, called matching rules. Listing 5.1 shows a possible definition in standard VPDL for such a view, with a simple matching rule.

To recap, the select part in VPDL is used to define which concepts and properties from which metamodel(s) will appear in the view (* means all properties). It also introduces new inter-model relations, i. e. the watched relation between User and Movie elements in our case. The from part allows users to declare the concerned metamodels, i. e. Movies and Users in our running example. Finally, the where part contains OCL-like expressions specifying matching rules for new inter-model relations, i. e. for watched in our case. In the example, we write a trivial rule that checks that among the tags of the movie (indicated as t, i. e. target of the possible link), there exists one whose name is equal to the value of the occupation for the user (indicated as s, i. e. source of the possible link). These OCL-like expressions are then automatically converted by EMF Views into an Epsilon Comparison Language (ECL) [2] matching rule that is used for computing the model view. It is worth mentioning that the letters "s" and "t" come from the syntax employed in EMF Views (inspired by the standard practice in ATL), where "s" denotes the Source and "t" denotes the Target of a relation. These variables are not defined directly in VPDL. Instead, they are defined in the generated ECL file.

2. https://www.eclipse.org/epsilon/doc/ecl/ *(Last Accessed in November 2024)*

As we said, this matching rule does not represent a real-world estimation. Moreover, the example already highlights important problems:

— A more realistic matching rule would require using a statistical programming library not available in VPDL.

— Engineers would have to use external tools to assess the rule's validity against real-world data.

In the following, we show our proposal to extend VPDL to define an automatic learning process for such matching rules from previous examples, effectively bypassing these problems.

5.3 Approach

5.3.1 Overview

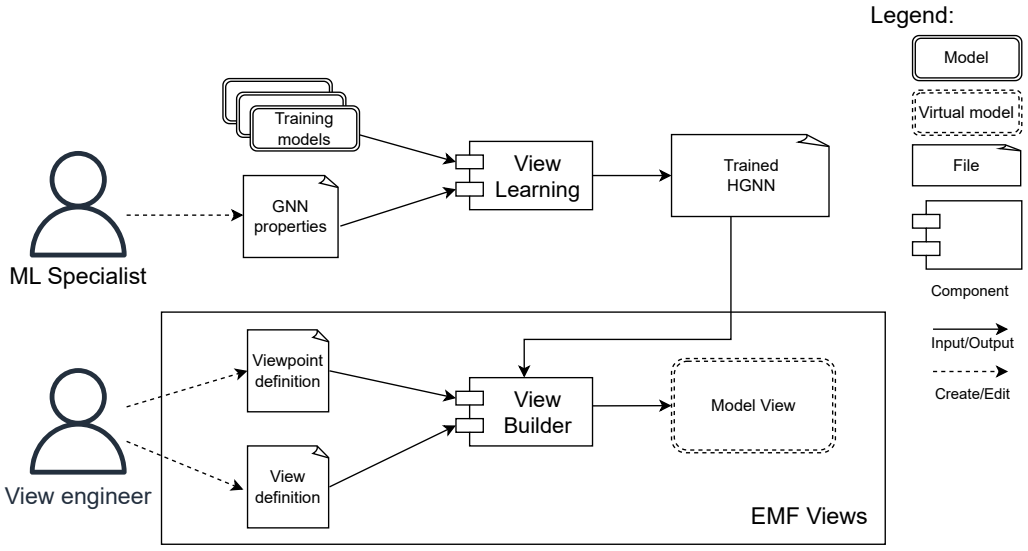

Figure 5.3 – Overview of the proposed approach

In the previous EMF Views approach (cf. the lower part of Figure 5.3), the view engineer has to provide two artifacts: a *Viewpoint definition* at the metamodel level and a *View definition* at the model level. These two artifacts can be partially generated from the VPDL specification. Then, the *View Builder* takes these two artifacts as inputs and builds a virtual model that materializes the specified *Model View*. In our extended approach (cf. the upper part of Figure 5.3), we complement EMF Views with a new *View Learning*

component to support the *View Builder* base component. A set of assignments for *GNN properties* is computed from the Viewpoint definition. It describes the architecture of the GNN and the hyperparameters for link prediction, including training and embedding. A *ML Specialist* can edit the value of these properties, e. g. to fine-tune the learning step. *Training models* are also required, including existing links used as examples for learning. Such existing models can come from different sources, e. g. they can be collected from legacy projects. Then, the *View Learning* component takes these two artifacts as inputs and generates a trained HGNN. The set of inter-model links are computed by the *View Builder* component using the trained HGNN, before constructing the corresponding view.

Note that the EMF Views already support delegating the computation of inter-model links to external tools since the links can be stored in a weaving model (cf. Shapter 2.1.1.4 for details). Hence, our proposed approach could reuse the standard structure of the *Viewpoint definition* and the standard *View Builder* component from EMF Views with no modifications. Moreover, the approach aims at decoupling the contributions of the *View engineer* and the *ML specialist*. Thus, the *ML specialist* can support the engineer by working on improving the accuracy and relevance of the inferred links without affecting the original *Viewpoint definition* and *View definition* made by the *View engineer*. Overall, we intend to make the use of DL as transparent as possible from the *View engineer* perspective. This way, they can focus solely on the modeling aspects while delegating ML integration and execution to our approach (and possible ML-specific optimizations to the *ML specialist*).

5.3.2 Extended ViewPoint Definition Language

We rely on the standard EMF Views for partially generating the *Viewpoint definition* and *View definition* from a specification in VPDL. Then, the View definition is manually completed to point to the actual resources, i. e., the contributing models. Additionally, our approach exploits our VPDL extension for generating default GNN architectures and hyperparameters (based on previous experiments) for the learning process.

```
1 create view watched as
2
3  select Users.User[id, name],
4       Movies.Movie.*,
5       Users.User join Movies.Movie
6       as watched
```

```
7
8  from 'http://paper/movies' as Movies,
9      'http://paper/users' as Users
10
11 where  "{s.id}<~s.movies~>{t.id, t.genres.value}"
12       for watched
```

Listing 5.2 – Extended VPDL for the running case using DL.

Listing 5.2 shows a snippet of our viewpoint specification in Extended VPDL for our running example. In this new version, the *create* and *from* parts remain unchanged. However, the *where* part no longer contains an OCL-like expression but a specific expression indicating, for each inter-model relation, the properties of the two models and the training relation to be considered for learning. It contains:

- A set of navigation paths starting from the source of the relation `s`, indicating the properties that should be considered for characterizing the source element. In our case, `{s.id}` indicates that the learning system will only use the `id` of the user (and not the name, age, etc.).
- A set of navigation paths starting from the target of the relation `t`, indicating the properties that should be considered for characterizing the target element. In our case, `{t.id, t.genres.value}` indicates that the learning system will use the `id` of the movie and the list of its `genres`. Note that the navigation expression can navigate the model to access attributes of other model elements, e. g. `Genre`.
- A navigation path indicating an existing relation used as the source of examples. This path is always represented between the two previous sets, with a specific arrow notation. In our case, `<~s.movies~>` indicates that the learning system will consider the `movies` relation as the set of examples to learn from (in the direction starting from `s`).

5.3.3 View Learning Component

Figure 5.4 shows how the new *View Learning* component is organized internally. This component realizes the bridge between EMF models and ML heterogeneous graphs. We create one heterogeneous graph per relation to learn. This graph is a bipartite graph that contains only connections between nodes from the source and target models. The bipartite graph that corresponds to a given relation is constructed in the following way:

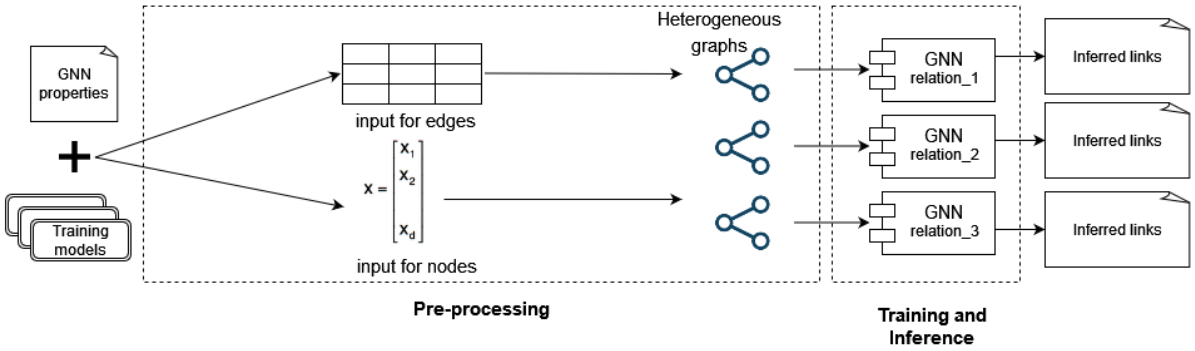

Figure 5.4 – Structure of View Learning and Inference

1. Embedding vectors (i. e. numerical representation in a lower dimension) for nodes are built by retrieving only the attributes involved in learning that relation (as indicated in VPDL) and by pre-processing them according to the GNN Properties.
2. An edge between two nodes is added if that relation in the training models connects those nodes.

In a second step, the component reads the *GNN properties* to instantiate a separate HGNN for each relation and performs training and inference. As shown in Listing 5.3, the *GNN properties* are serialized in a JSON file split into information blocks containing all necessary parameters for the HGNN definition, embedding, and training. We opted for the JSON format to allow for a straightforward modification of these parameters by the *ML specialist*.

The core elements of the HGNN, detailing its message-passing mechanisms and aggregation operations, are encapsulated by the `ARCHITECTURE` label. Each key within this block corresponds to a single aspect of the HGNN's configuration: The `OPERATOR` key denotes the type of layer used for aggregation, while `CONVOLUTIONS` indicates the layer count. The `ACTIVATION` key specifies the activation function employed between layers, and the key `HIDDEN_CHANNELS` represents a numerical value determining the feature dimensions within hidden layers. Furthermore, the `CLASSIFIER` key indicates the function utilized to compute the final likelihood score for graph edges (i. e. edge decoder).

The usual hyper-parameters for standard NN tasks as encapsulated by the `TRAINING_PARAMETERS` block, including epochs (`EPOCHS` key), learning rate (`LEARNING_RATE` key), and specific parameters for the link prediction task (e. g. addition of negative edges during training and strategies for edge splitting in training-test-validation). Additionally, this block

3. A LINKS_PATH property can be added when inter-model links are serialized in separate files.

```
{
  "watched": {
    "ARCHITECTURE": {
      "OPERATOR": "SAGEConv",
      "CONVOLUTIONS": 2,
      "ACTIVATION": "relu",
      "HIDDEN_CHANNELS": 64,
      "CLASSIFIER": "dot_product"
    },
    "TRAINING_PARAMETERS": {
      "EPOCHS": 2,
      "LEARNING_RATE": 0.001,
      "ADD_NEGATIVE_TRAINING": false,
      "NEG_SAMPLING_RATIO": 2.0,
      "TRAINING_SPLIT" : 0.1,
      "VALIDATION_SPLIT" : 0.1,
      "SOURCE_MODEL_PATH": "users.xmi",
      "TARGET_MODEL_PATH": "movies.xmi"
    },
    "EMBEDDINGS": {
      "s.id": "id",
      "t.id": "id",
      "t.genres.value": "enum"
    }
  }
}
```

Listing 5.3 – GNN properties JSON file

encompasses the file paths for the *View Learning* component, namely the serialized models utilized during training. In case *training links* are stored in an independent file, this can be given too in an additional property (`LINKS_PATH`). Since we consider the inter-model link identification as a link prediction problem, it makes sense to split the graph into links (i. e. the training/validation sets are split based on the links). Being so, the keys `ADD_NEGATIVE_TRAINING` and `NEG_SAMPLING_RATIO` are related to different strategies during this split, specifically regarding the inclusion or not of negative edges and the ratio for its inclusion. `TRAINING_SPLIT` and `VALIDATION_SPLIT`, as the name suggests, define how the links are split into training and validation sets, respectively. The user gives the paths to the models used for training through the keys `SOURCE_MODEL_PATH` and `TARGET_MODEL_PATH`.

The `EMBEDDINGS` block lists the properties specified for that relation in VPDL. For each one of them, we select their corresponding encoding scheme. For encoding, we currently support the following:

— `id` - Encoded as a lookup table for the element/node;
— `enum` - Encoded as a set of a fixed list. The strategy used is one-hot encoding;
— `string` - Uses a pre-trained language model to represent strings numerically. The user can indicate which model to use from the SentenceTransformers [270] library. [4] The current implementation is limited to the use of the SentenceTransformers library. However, the use of other libraries, including MDE-specialized language models [82], are also possible with few adaptations;
— `number` - The value is cast to a float representation.

The definition of optimal default parameters is a problem that depends on various factors: the task you are working on, the characteristics of your graph data, the GNN model's architecture being used, etc. Our work does not explicitly discuss the definition of criteria for these default parameters. However, our approach is designed to simplify the modeler's work and leave these parameterization tasks to the ML specialist. Indeed, the ML specialist is more likely to have the necessary domain knowledge to make informed decisions.

It is also worth mentioning that some parameters, such as dataset split (e. g., 80 % for training and 20 % for validation) and the initial learning rate (often set to 0.001), are somewhat standardized within the ML community.

4. `https://www.sbert.net/docs/pretrained_models.html` *(Last Accessed in November 2024)*

5.4 Implementation

This section describes the implementation of a prototype supporting the proposed approach. This prototype is open-source and publicly available. [5] We notably used the EMF as a basis of EMF Views and its *View Builder* component, and Xtext [6] for the modification of VPDL. The *View Learning* component, mapping EMF models in a Python context and allowing different HGNN architectures, requires the use of the PyEcore [7] and PyTorch Geometric [8] Python libraries, respectively.

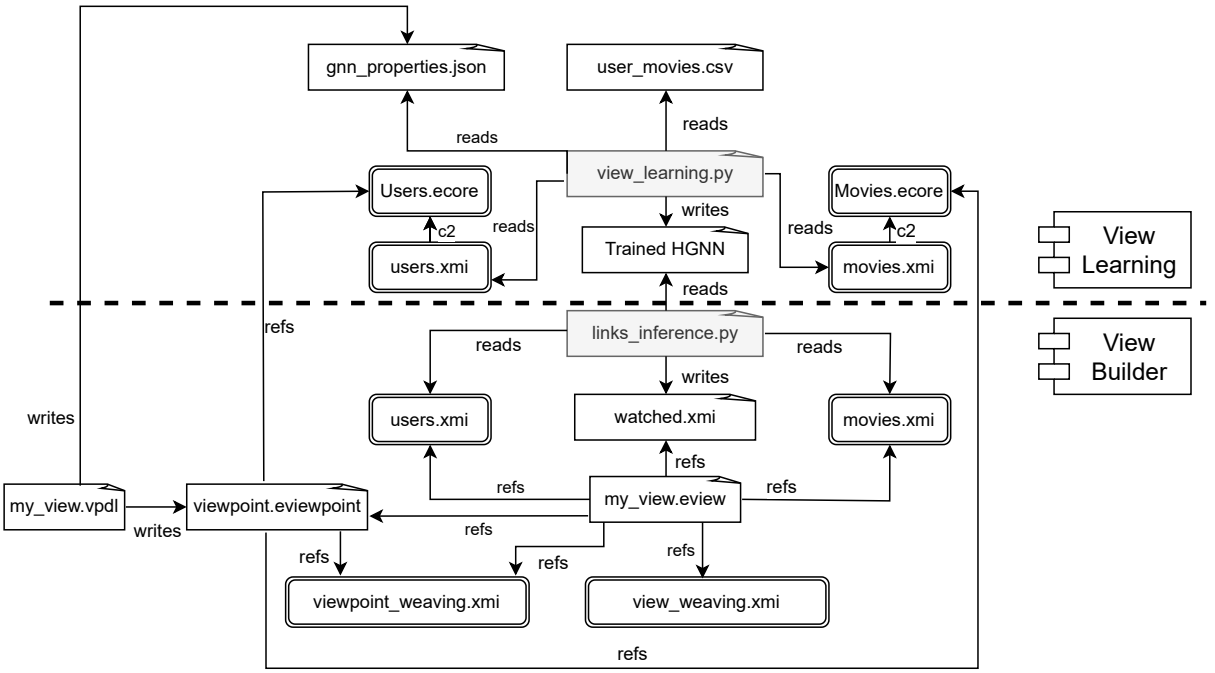

Figure 5.5 – Main files in the prototype, applied to the running example (c2=conforms-to; refs=references)

Figure 5.5 shows the essential files and their organization in the implemented prototype, applied to the running case. *Users.ecore* (the source of the link) and *Movies.ecore* (the target of the link) are the two metamodels considered in our viewpoint. *Users.xmi* and *Movies.xmi* are the two models that conform to the two previously mentioned metamodels (respectively) and that contain the actual data to build the target view. In practice,

5. `https://github.com/NaoMod/Support-ML-Relations-Model-Views` *(Last Accessed in November 2024)*
6. `https://www.eclipse.org/Xtext/` *(Last Accessed in November 2024)*
7. `https://github.com/pyecore` *(Last Accessed in November 2024)*
8. `https://pytorch-geometric.readthedocs.io/` *(Last Accessed in November 2024)*

the view engineer writes in *my_view.vpdl* the VPDL specification of her viewpoint. The *viewpoint.eviewpoint* descriptor file is automatically generated from the VPDL specification. It contains pointers to the contributing metamodels and the EMF Views internal weaving model at the viewpoint-level [42]. The *my_view.eview* descriptor file is also automatically generated. It contains pointers to the corresponding viewpoint descriptor file, the contributing models, and the EMF Views internal weaving model at the view level. In parallel, the ML specialist checks and adapts accordingly the *gnn_properties.json* complementary JSON file provided by default. Once done, they also provide the *user_movies.csv* inter-model relation file that is used by the *view_learning.py* Python code to build and train a proper *HGNN.model*. In addition, this Python code also produces a performance evaluation chart (cf. section 5.5). Finally, the *relation_inference.py* Python code uses this trained ML model to generate the *watched.xmi* inter-model links file. This newly generated file can be ultimately used in the target view. To summarize, the *View Learning* component in our extended approach is implemented in these two *view_learning.py* and *relation_inference.py* Python files.

5.4.1 Limitations

On the VPDL side, we support the inference of several inter-model relations, each learned by accessing an arbitrary set of properties of the training dataset.

For GNN properties, we currently support only a specific set of values in the `ARCHITECTURE` section:

— The `OPERATOR` is either “SAGEConv” or “HANConv”;
— The number of `CONVOLUTION` layers is fixed to 2;
— The `ACTIVATION` function is either “relu” or “tanh”;
— The `CLASSIFIER` is systematically “dot-product”.

The CSV file containing relations from previous models is considered as existing information adapted from legacy projects. All these elements will be progressively improved in the following versions of our prototype (cf. Chapter 7 for a complete and detailed list of future perspectives).

5.5 Evaluation

To evaluate our approach, we consider three main aspects: 1) the efficiency of the HGNN for link prediction on real-world models, 2) how our approach compares to traditional solutions in terms of the number of Lines of Code (LoC) necessary to obtain an identical result, and 3) the ability to learn pre-determined attribute-based matching rules. For 1) and 2), we apply our solution to the running example depicted in section 5.2.2. For 3), we consider simple views on randomly generated models conforming to minimal metamodels and containing different relations.

5.5.1 Evaluation on the Running Example: Prediction Accuracy

To evaluate that our solution works efficiently enough for the running example, we created instances of the Users and the Movies metamodels (see Figure 5.1), using data from the *MovieLens* dataset provided by the GroupLens research lab [266]. The first column of Table 5.1 presents the figures from the *ml-latest-small* dataset,[9] including the time spent for training the HGNN.

	MovieLens	**AB**
nodes	610 Users, 9742 Movies	1000 As, 300 Bs
edges	100 863	5000
training time	58 s	∼50 s

Table 5.1 – Dataset and training figures for MovieLens and AB

The evaluation metrics selected for the problem are the Receiver Operating Characteristic (ROC) curve and the respective area under the curve (AUC_ROC). The AUC_ROC measures the ability of a GNN to distinguish between different classes by plotting the true positive rate against the false positive rate [271]. A value close to one denotes a good link prediction accuracy.

Figure 5.6 shows the resulting ROC curve for the `watched` relation described in the section 5.4, for different values of the threshold. We can observe an $AUC\_ROC >= 0.9$, denoting good accuracy for this example.

9. `https://grouplens.org/datasets/movielens/latest/` *(Last Accessed in November 2024)*

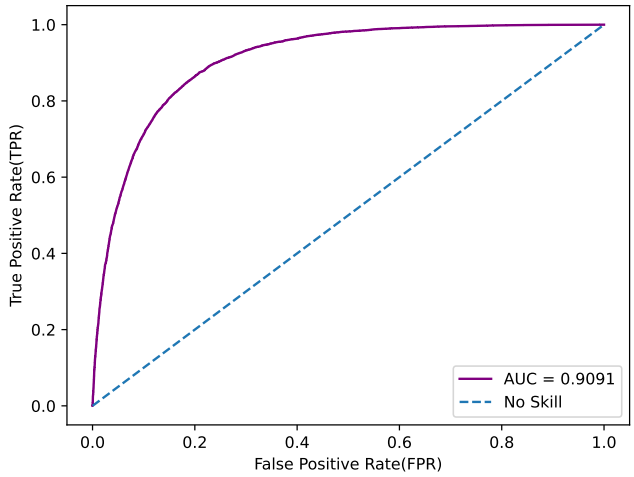

Figure 5.6 – ROC curve for the running example

5.5.2 Evaluation on the Running Example: LOC

With our approach, view engineers can now specify our example model view via concise statements in the extended VPDL and possibly modify a JSON file to fine-tune the DL configuration. Indeed, our running case only involves 39 LOC, consisting of 9 lines of VPDL code and 30 lines of a partially generated JSON configuration file.

We asked a proficient Python/PyTorch programmer to write an equivalent program with the same input files. This resulted in 385 [10] lines of Python code (excluding blank lines and comments). As we stated, our approach's main objective is to hide the use of DL code as much as possible so that engineers can focus on their modeling activities.

5.5.3 Learning Different Matching Rules

We consider a different example from our running one to evaluate this metric. Figure 5.7 shows the two metamodels (also in Ecore), each containing one metaclass. The metamodel named *Left* has a class `A` with one numerical attribute `a` and one string attribute `s` besides its identifier. The metamodel named *Right* has only a class `B` with three numerical attributes `b`, `c` and `d`, and one string attribute `s`. Finally, a relation called `allBs` relates class `A` to class `B`. This example is named the *AB example*.

In this experiment, we evaluate the capacity of our tool to learn given matching rules on the *Left* and *Right* metamodels. We start defining three matching rules:

— $A.a = B.b$, i. e. two elements are related if they have the same numerical attribute

10. Available in the project repository

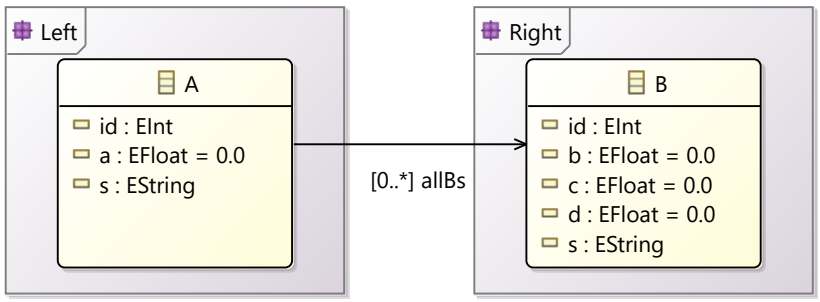

Figure 5.7 – Metamodels for the AB example

— $A.a = B.c * B.d$, i. e. two elements are related based on a simple mathematical operator (integer multiplication)
— $A.s.contains(B.s)$, i. e. two elements are related based on a simple string operator (containment).

Note that these matching rules do not consider the structural aspects of the model. We evaluated the capability of the HGNN to learn from attribute values per node type. Given one of the three matching rules, the sample models are generated with random values. Still, the generator guarantees that all instances of `A` have at least a matching element of type `B` for that matching rule. Finally, the generator connects a random ratio of matching elements, leaving many missing links, i. e. connections that should exist but were not created. Once the data set is created, we follow the same approach explained in the previous sections.

Figure 5.8 shows one ROC curve per matching rule. These curves show promising results ($0.85 <= AUC\_ROC <= 0.94$) considering the small amount of data we used (5000 connections for 4000 nodes). They show that, by using standard hyperparameters for HGNN definition and training, our tool can learn simple matching rules on attribute values.

5.5.4 Summary of the Results

The ROC curves in Figures 5.8 and 5.6 show a promising efficiency in terms of relation inference. Aiming to show the improved stakeholders' productivity while working in a model views environment, the section 5.5.2 presented some figures on LoC written by them, highlighting our contribution to simplifying and expediting tasks with model views, mainly related on the definition of inter-model relations. In the AB example, the results are helpful for the user despite the relatively small size of the data set. The AUC_ROC

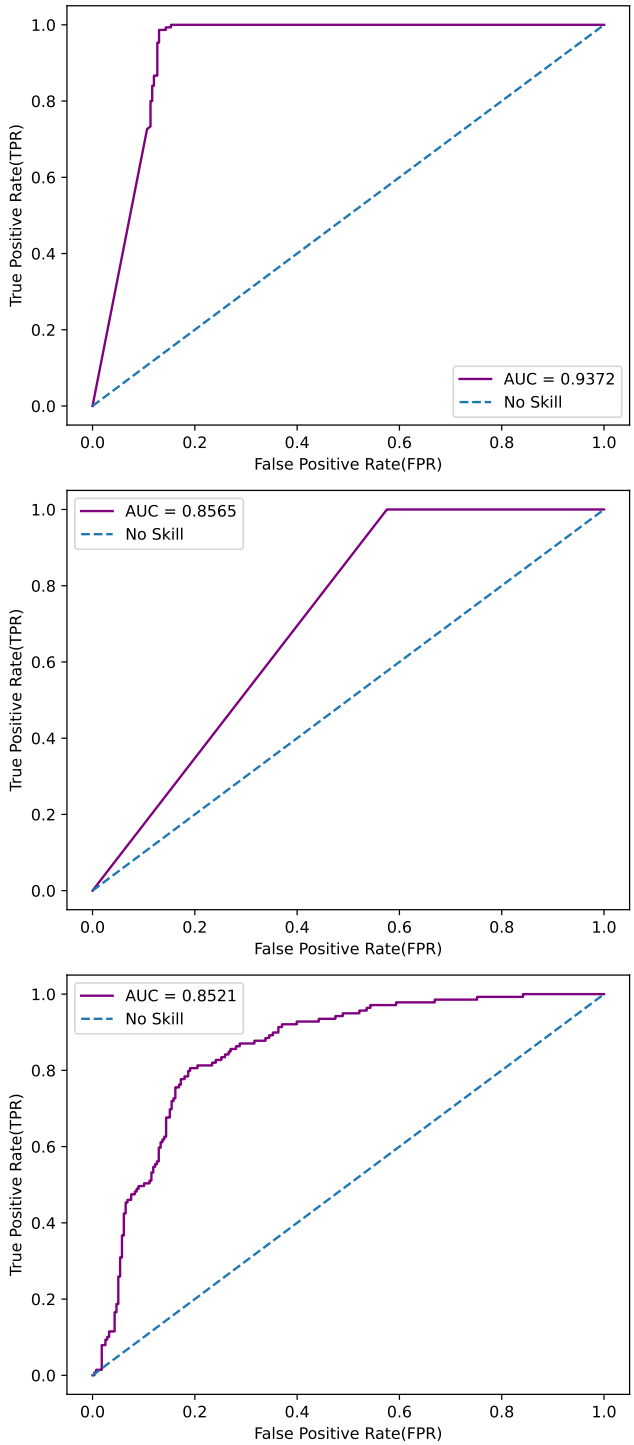

Figure 5.8 – ROC curves for different AB relations: $A.a = B.b$ at the top, $A.a = B.c * B.d$ in the middle, and $A.s.contains(B.s)$ at the bottom

values obtained in our experiments indicate the ability of our approach to learn structural relations between model elements accurately for at least one case (running example) and to learn attribute-based relations significantly better than random-guesser in other cases (AB example). The approach relies on the expertise of ML specialists to optimally parameterize the architecture and learning process of the HGNN for the given task.

5.6 Conclusions

In this chapter, we proposed an approach for automatically inferring inter-model links in the context of model views. This approach relies on DL techniques, particularly on HGNNs. Its objective is to support view engineers in specifying viewpoints and corresponding views when inter-model links can be inferred from existing examples. The proposed approach intends to lower the barrier of using DL in the context of model views. It also facilitates collaboration with ML specialists who can help the view engineer improve the link prediction accuracy.

To this end, we refined and extended the existing VPDL model view specification language from EMF Views to properly integrate the automated generation and use of view-dedicated HGNNs. In practice, we implemented our approach by combining the EMF Views solution with two Python libraries, PyEcore and PyTorch Geometric, dedicated to model handling and HGNNs, respectively. Based on this implementation, we conducted a first set of experiments that showed promising results concerning the automated inference of inter-model links, reducing the number of DL code to be written by engineers.

The related work (cf. section 2.2.2) shows that GNNs appear to be particularly adapted to such scenarios where additional data must be inferred from existing models, notably related to their structural aspects. Compared to these research efforts, we are addressing a more limited problem, computing the probability of the existence of a particular link. This allows us to use HGNNs, a solution that showed high performance in this specific problem. More aligned with our proposed approach, we can complement the related works Chapter with the ones looking into preserving the consistency and synchronization of a given model view. Guerra *et al.* propose a formal approach for specifying the relations between modeling languages through the use of a pattern-based language for inter-modeling with a focus on model-to-model transformations, model matching, and model traceability [272]. Although their approach is not explicit about model views, their definition of inter-model links can be considered a potential mechanism for computing views based on patterns.

Quite differently, the ModelJoin solution [201] proposes to rely on a metamodel generator and higher-order transformations to compute these links on the fly. To the best of our knowledge, AI(namely HGNN) techniques have not yet been directly used in such a model view context.

Part III

Conclusions and Future Work

Chapter 6

Conclusion

In this chapter, we summarize and wrap up the thesis's primary objective: help engineers creating model views supported by AI-augmented solutions, mainly through the use of DL techniques. To do so, we sum up the targeted problem, recall our corresponding research questions, and answer each before concluding with the thesis's overall contributions.

Model view solutions are designed to assist engineers in understanding and working with complex systems, such as CPSs. They provide mechanisms for model federation, i. e. a straightforward approach to combining, filtering, and augmenting original models to address specific objectives. However, creating these views involves significant challenges.

First, the process is labor-intensive at *design time* (i. e. when engineers analyze the problem to define the views). It demands expertise in view creation mechanisms and a deep understanding of the contributing models, which can be time-consuming and error-prone. Second, at *runtime* (i. e. during the computation of view elements after the views are defined), complex problems often necessitate specialized mechanisms to derive the final view. For example, establishing links between contributing models (i. e. inter-model relations) can be particularly challenging when no clear matching strategy exists.

Given that model views are a critical component of MDE (also MBE) workflows, this thesis aims to address these challenges and simplify the creation of model views. The adoption of widespread DL techniques (namely LLMs and GNNs) aligns with recent advancements in the MDE community, where these techniques are increasingly used to tackle complex problems (cf. section 3.2). We expect the proposed improvements will facilitate adopting and effectively using model view solutions for CPS projects, such as those in the AIDOaRt context [29] that demonstrate the need for AI-augmented model view solutions. For instance, the VCE use case that demands it for efficient and automated methods to address information fragmentation and heterogeneity within their MBSE toolbox (cf. 2.3).

As a solution, we proposed a twofold approach to enhance the usability and automation of model views by leveraging DL techniques during design time and runtime. Given the

effectiveness of DL in addressing various SE and MDE challenges, we adapted well-known techniques to meet the specific needs described above. Our approach relies on applications of NNs, the *de facto* standard in the ML community for DL.

Firstly, we used LLMs (i. e. NNs based on the transformer architecture, primarily oriented toward textual tasks) for automated view specifications. Our contribution (cf. Chapter 4) provides an in-context approach to using off-the-shelf LLMs for this task. and HGNNs for inter-model link computation. We demonstrated its feasibility and discussed the benefits of PE, tool-augmentation, and problem decomposition in applying LLMs to a particular MDE problems. Following our established methodology, this implementation can help us answer the **RQ1** (cf. section 1.2).

Answer to RQ1

Considering the model view solutions that based the description of the viewpoint and view (even partially) on textual DSLs, the use of LLMs is a good strategy since this DL technique is especially relevant dealing with text and code. It is highly adapted to deal with information at the metamodel level. Additionally, the off-the-shelf LLMs can include long textual information with textual serialization of the metamodels involved in the view creation. Our prototype using EMF Views demonstrates that evaluating LLMs alone is insufficient. Complementary assessments involving PE and tool augmentation are also essential.

The second contribution (cf. Chapter 5) addresses the particular runtime challenge of automating the computation of inter-model relations in model views. We enhanced the definition language (i. e. VPDL) to enable an efficient use of HGNNs (i. e. NNs designed for heterogeneous graph data) to infer these links through link prediction capabilities. We demonstrated an efficient solution when the language definition lacks expressiveness by adapting a well-known ML problem to the MDE domain. Supported by the research on related work (cf. Chapter 3) and by our prototype and evaluation, we can answer the **RQ2** (cf. section 1.2).

Answer to RQ2

Different model federation techniques rely on underlying techniques that store links between model elements. This is also the case for some model view solutions, e. g. EMF Views that rely on weaving models to cope with this. On the model level, DL techniques can be handy in dealing with the problem of finding the links between models (i. e. link prediction). This is especially true when considering their heterogeneous structures. We showed that using HGNNs gives promising results. Our implementation also helps ML specialists effectively participate in the MDE workflows when using model views. Given the probabilistic nature of DL algorithms, these specialists' contributions may help achieve better results.

Both contributions were implemented within EMF Views, a model view solution designed for creating model views over EMF-compatible models. This allowed us to validate benefits and limitations through the evaluation of the prototypes [90], based on selected examples and metrics, helping in answering **RQ3** (cf. 1.2). The examples were selected mainly from the literature but also combined with experiments using data collected from AIDOaRt partners (cf. section 2.3.5 for a better description of the data).

Technically, for the first contribution, we applied and extended LangChain tooling together with PyEcore to handle EMF models and use LLMs to draft VPDL code (the EMF Views DSL for view definition). For the second contribution, we used PyTorch Geometric to implement HGNN algorithms (e. g. GraphSAGE) and applied their link prediction capabilities in EMF views examples. We extended the VPDL language to enable view engineers to use HGNNs (also homogeneous GNN) when needed effectively.

Answer to RQ3

We consider that our prototypes and their respective evaluation provided in the contributions chapters (4 and 5) form a step further in showing how to use DL to improve model view solutions, an already identified challenge in adoption this kind of tooling by MDE practitioners. Although it demands developing (and consequently integrating) specialized tools in an already complex workflow and lacks reliable datasets (both for training and evaluation), we believe that its advantages are worth the effort. From the main advantages of relying on DL (and even broader ML), we can highlight i) the potential application on edge cases (e. g. finding links when matching rules are not expressive enough) and ii) the efficiency in its application to ease the model views creation (e. g. to reason over metamodels when the engineers do not have enough resources to understand them deeply).

The novelty of this thesis lies in its dual approach, which applies DL techniques to model view problems both at design time and runtime. Traditional model view solutions focus on manual methods or rule-based systems. By integrating LLMs and GNNs, this work not only automates significant parts of the model view specification and link inference processes but also broadens the applicability of DL in MDE in general. Additionally, our implementations facilitate the involvement of ML specialists in the model views pipeline, enabling multi-stakeholder participation. In a broader sense, our contributions also validate the utility of AI-augmented model views in support of the CPS development. To the best of our knowledge (cf. Chapter 3), this is among the first works to leverage these techniques for model view challenges, positioning it as a novel contribution. Furthermore, other strategies for model federation (e. g. model merging, megamodeling, etc.), even being different ways of providing federation, can potentially benefit from extensions and adaptations of our approaches. A discussion on the open challenges and future perspectives for our work is given in Chapter 7.

Chapter 7

Future Perspectives

This chapter addresses the limitations identified in our work and proposes potential solutions for each, opening questions and providing narrowed and global future perspectives. Our discussion emphasizes the application of LLMs and HGNNs within MDE while acknowledging challenges in developing sustainable DL solutions in this field.

7.1 Future Work

Considering that we have a twofold contribution, we can also split our future work into two parts: one to cover the potential improvements on using LLMs at the design time of model views and another to investigate the possible improvements of our use of (H)GNNs at runtime.

7.1.1 Using LLMs for Model Views

Evaluation and Benchmark Datasets To overcome the lack of datasets for evaluating our LLM-based approach, broader model view datasets are necessary. Currently, we rely on examples within the EMF Views and ATL transformation domains, which limits generalizability across other MDE applications. In practice, we could experiment with our approach and implementation on a more extensive dataset encompassing diverse modeling contexts. Such datasets would benefit benchmarking efforts and support reproducibility. We can grab inspiration to deal with it from research efforts like the ModelSet [259] and the ModelXGlue [273], for example.

Exploring Additional Prompting Techniques The work on the prompt templates themselves could be continued without altering the rest of the approach and implementation (cf. Chapter 4). As we have observed, PE is an empirical discipline often requiring many trial-and-error iterations. Refining our prompt templates allows us to explore nu-

merous strategies to enhance results. For instance, prompt templates could be tailored to better capture the semantics of the target metamodels by incorporating domain-specific terminology and changing structured representations (e. g., leveraging tabular formats or JSON-like syntax for metamodel serialization instead of PlantUML). Additionally, iterative prompt tuning [274] could be applied, where feedback from intermediate outputs is used to refine the prompts during execution dynamically. Complementary to this, exploring other PE techniques, beyond the current use of CoT and few-shot prompting, offers different ways for potential improvement [193, 275–278]. Also, during the prompting process, RAG [257, 279] could be integrated to include partial models and external knowledge bases of model specifications. Similarly, self-consistency decoding [280] could be investigated to address the variability of LLMs's outputs, ensuring the generation of robust and reliable model view specifications. All these enhancements could be systematically studied and evaluated to identify their impact on automating and improving the creation of model views.

Improving with LLM-Based Agent Architectures To improve our architecture, we can shift from a single LLM-agent system (cf. section 4.3) to a multi-agent paradigm. This involves specialized agents focusing on distinct tasks, like model querying and semantic mapping, improving efficiency and accuracy in describing model views. For example, one agent could focus on model querying for data extraction from the metamodels, potentially using fine-tuned LLMs [241]. Another agent could handle semantic mapping ensuring that the extracted data aligns accurately with the user intendeds. To do so, we can get inspiration on other ML reserach that deals direct with the mapping between heterogeneous enviromments [281]. Additional agents could be designed to assist with specific subtasks such as consistency checking, model validation, or generating explanations for the inferred results to be incorporated into the draft given to the EMF Views user. These specialized agents could collaborate and exchange intermediate outputs to refine their results iteratively, leveraging Multi-Agent Collaboration (MAC) principles [282], strategy that is rapidly gaining traction in the AI/LLM communities [283]. Moreover, the interaction between agents could be augmented with advanced coordination mechanisms, such as using a controller or orchestrator agent that dynamically assigns tasks and manages dependencies among the agents. Such a multi-agent approach could be systematically studied to assess its impact on improving the automation and reliability of model view description generation. Since our current prototype is implemented using LangChain, the

implementation of a MAC though the use of LangGraph [1] is a straightforward first step for prototyping.

Integrating Human-in-the-Loop In addition to improving the quality of the results, a direct benefit of a multi-agent approach could be better integration of the human in the loop [284]. Indeed, our current approach is voluntarily designed to allow engineers to provide only the initial inputs (i. e. the Ecore metamodels and a relatively small description of the desired view). It then works as a black box until the model view definition is generated as output. In a possible alternative version of our approach, we could i) collect intermediate inputs from the engineers (e. g. in chat mode) and ii) consider these inputs for the different internal iterations regarding the three main parts of the view. Thanks to such a more interactive approach, we may obtain comprehensive model view definitions that better correspond to the engineers' wills.

Enhancing Tooling Integration The current implementation of our LLM-based application could be upgraded regarding provided tooling features. Notably, we could work on improving the direct integration with EMF Views inside Eclipse. In the current version, the generated model view specifications in VPDL must be manually copied in the EMF Views/Eclipse environment. Automating this step would enhance productivity and usability. To address this, we could design and implement a seamless interface between our Python-based implementation and the Eclipse workbench hosting EMF Views. Such integration could be achieved by developing a dedicated plugin (e. g. extension point) that acts as a bridge between the two environments using the EMF APIs, providing e. g. automatic import of VPDL drafts or live updates. This evolution would bridge the gap between our research prototype and a production-ready solution.

7.1.2 Using GNNs for Model Views

Improving Evaluation Metrics On the evaluation aspect, our approach can benefit from incorporating a comprehensive analysis of the main HGNN default parameters, including possible integration with other ML automation techniques (i. e. AutoML) for hyperparameter optimization. Another improvement for future work includes evaluating different examples from the MDE domain, including when these models have a higher semantic gap between them, distinct from our examples. In a different direction, future

1. `https://www.langchain.com/langgraph` *(Last Accessed in November 2024)*

work can also include using different DL techniques to complement other aspects of the definition and use of model views, enabling comparisons between the use of GNNs and other potential approaches.

Evaluating the Learning Capacity of GNNs We showcased the expressiveness of our language and approach by implementing a few model views with different inter-model relations. However, additional experiments are still required to systematically assess which matching rules could be effectively replaced by a trained HGNN. Moreover, we plan to experiment with our approach more generally in the context of inferring matching rules within model transformations since they are also valuable for model views development, as we demonstrated in the LLM contribution (cf. sub-section 4.5.2). Indeed, during the work of this thesis we already started to create a prototype in this direction. [2] Finally, the perspective of a view dataset/benchmark mentioned before in sub-section 7.1.1 can also be helpful for the HGNN experiments.

Designing a DSL for GNNs Our Extended VPDL and its companion files represent the initial version provided with our implementation. While these components enable basic functionality, there is significant room for improvement in terms of usability, expressiveness, and integration. The next step is to develop a dedicated DSL for GNN properties, replacing the presented JSON file (cf. section 5.3), and integrate it more smoothly with VPDL. Such a DSL could feature a declarative syntax that allows users to specify GNN architectures, link prediction configurations, and training parameters in a concise and human-readable format. Additionally, the DSL could include, for example, annotations that could be used to link specific GNN configurations to view elements or metamodel components, ensuring a seamless flow between the DSL for GNN properties and the overall model view specification process. To achieve these goals, a study of existing work related to DSLs for supporting ML activities [124, 285, 286] can guide our future implementations. Furthermore, to ensure the integration of this new DSL with VPDL, we could explore mechanisms for interoperability, such as embedding GNN-specific constructs directly within the VPDL syntax or providing a seamless mapping between the two languages. Overall, developing a dedicated DSL for GNN properties represents a significant step forward in enhancing the usability and adaptability of our approach.

2. `https://github.com/jameswpm/transformations-emf-views` *(Last Accessed in November 2024)*

Supporting Additional (H)GNN Architectures While the current GNN properties allow us to effectively address the running and synthetic examples used for evaluation, more options to describe GNNs and training processes could help with edge cases. For example, while we support the aggregation function selection, we do not support standard internal parameters for each function, e. g. learning additive bias, application of linear transformation after activation layers, normalization of output, etc.. These values can vary among different aggregation functions. Such parameters can exhibit considerable variability depending on the specific function or task requirements, highlighting the need for a more comprehensive and flexible configuration approach. By introducing them in the description of GNN architectures, we could support a broader range of use cases, including edge cases that might fall outside the capabilities of our current implementation.

Improving the inference capability of the approach HGNN performance could be optimized through offline hyper-parameter tuning or exploring alternative architectures. We may allow users to choose among threshold selection strategies (e. g., based on the provided AUC_ROC metric or adding new complementary metrics [287]). These improvements will sometimes require further extending VPDL. For instance, we may want to support the user's meta-path specification to guide the discovery of complex relations [288]. Meta-paths are sequences of node and edge types that define meaningful relationships within heterogeneous graphs. Additionally, experimenting with alternative DL strategies, such as incorporating attention mechanisms or leveraging message-passing strategies tailored to specific data structures, could open up new possibilities for improving inference quality. This would require adjustments not only to the VPDL syntax but also to the runtime mechanisms responsible for parsing and interpreting these specifications.

7.2 Global Perspectives

This section considers broader perspectives potentially applicable to our contributions and other DL applications to model views and MDE in general. This analysis is important given the potential improvements brought by this combination to develop complex systems like CPSs.

It is worth mentioning that this thesis can be broadly classified into the called "AI/ML for SE", considering MDE as a SE paradigm [289], which means use AI to solve MDE problems. However, recent research efforts appeared in the trend of "SE for AI/ML." [290]

and so "MDE for AI." [291], using model-based techniques in the the engineering of AI-powered solutions. In the descriptions below, we consider both research streams as potential extensions and further discussions over our contributions.

Testing our approaches and implementations with other languages Our implementations consider model views created for EMF Views as the main case study. Consequently, our tests were limited to using VPDL as the main DSL for view definition. Although we also conducted preliminary experiments with partial solutions using ATL (cf. section 4.5), the diversity of DSLs employed for creating model views, as highlighted in related work (cf. section 3.1), suggests broader applicability. To address this diversity, extending or adapting our solutions to support other model views contexts is a promising direction. This could involve tailoring our approach to work with other DSLs, exploring languages that offer distinct mechanisms for model querying and different views manifestation. Furthermore, we see potential applications of our approach beyond modeling languages. For instance, similar patterns could be leveraged in general-purpose query languages such as GraphQL, where concepts like filtering, selection, and transformation align with core aspects of model views.

Exploring synergies between GNNs and LLMs Our contributions in the current state are split between using LLMs for the design phase and HGNNs for the runtime, respectively, leveraging textual and graph representation of models. Recent initiatives [292] proposed the use of GNNs to enhance pre-trained LLMs using grounded knowledge, improving RAG applications. In complement, the use of graph-based strategies to complement LLMs [293] is also trending, mainly through the use of Knowledge Graphs (KGs). We showed the capabilities of LLMs in understanding the base models used in model views in combination with GNNs to execute link prediction, so it is a natural way to continue our work combining these techniques. For example, one first step in this direction is to change our LLM-application to generate the VPDL already including our defined new syntax for the GNN-powered recommender. This will enable the automation between the design and runtime EMF Views workflow.

Improving Scalability and Performance Some optimization techniques could be applied to enhance the scalability of our approaches. For LLMs, the use of self-hosted models can enable strategies such as pruning [294], quantization [295], or knowledge distillation [296] to improve inference time, enabling their deployment in resource-constrained

environments (especially important for CPSs). For GNNs, parameter reduction techniques, such as dimensionality reduction of node embeddings, could significantly lower computational overhead while maintaining model accuracy [297]. In addition to optimizing individual components, they could be coupled with caching mechanisms to reduce redundant computations when the same or similar model views are queried repeatedly. Future work could also investigate adaptive sampling techniques for large-scale industrial applications, such as those addressed by AIDOaRt. Moreover, profiling and benchmarking the runtime performance of our approach under increasing data loads could guide targeted optimizations, such as hybrid processing strategies that combine on-device and cloud-based computation. Finally, reusing the strategy applied by Bruneliere *et al.* increasing the model size will enable us to test our approach up to large-scale models [42].

Considering Safety, Bias, and Ethical Problems As LLMs become more pervasive, their ethical implications have garnered attention, especially concerning bias and safety issues [298]. LLMs can unintentionally perpetuate stereotypes or generate harmful content, depending on their training data and the team that trained it [299]. Research efforts focus on mitigating these risks through better data curation, fine-tuning, and RL strategies, but challenges remain. In the context of model views, as proposed in our contribution, as LLMs are used to generate or suggest model viewpoints, the biases could inadvertently lead to skewed or incomplete model views, negatively impacting decision-making, model analysis, and system design. LLMs also pose risks when used for misinformation, spam generation, or even malicious purposes, calling for the establishment of robust guardrails and governance frameworks for their deployment [185, 189]. We could adopt strategies such as bias detection and mitigation techniques tailored to model view generation to cope with these risks. These techniques might involve better data curation and incorporating RL strategies that reward the generation of diverse, balanced, and ethically sound model views. Moreover, we can introduce manual review stages in the model view generation process to ensure that harmful or biased outputs are identified and rectified before they impact engineering processes.

Appendix A

Prompts used for LLM experiments

A.1 Introduction

This appendix intends to present the extra prompts used in our LLM-based application described in Chapter 4. Firstly, we present the two other prompts used in our provided tool, which means both the prompts used for the SELECT block and the WHERE block of the VPDL file. They complement the one used for the JOIN block already presented in the referred chapter. These prompts are given as-is to the EMF Views user. It is unnecessary to make any extra edits to perform likely the results presented in the evaluation (cf. Section 4.5). We also present the prompt used to perform the evaluation within the LLM-as-judge strategy.

A.2 Extra prompts

The listing A.1 presents the prompt template for the SELECT block. Similarly, the listing A.2 presents the prompt for the WHERE block. Their structure and phrasing are also based on the CoT PE strategy, already explored and explained in the Chapter 4. As explained in our contribution chapter (cf. Chapter 4), they complement our approach's implementation and are already adapted to be used in a LangChain workflow.

```
"""
You specialize in reason on PlantUML metamodels, especially
    selecting and filtering each class's attributes.

Given two metamodels and a list of relations containing classes'
    pairs, your task is to select a set of attributes for the
    metamodels' classes.

An attribute should be selected if it is unique among the two
```

```
    classes in a relation or if it is a collection that contains
    one of the classes in the relation.

For the input relations list, you may assume the following
    template:

{{
    "relations": [
        {{
            "name": "relationName",
            "classes": ["class_name_from_first_metamodel", "
                class_name_from_second_metamodel"]
        }}
    ]
}}


{format_instructions}

When generating the response, you should follow these rules:
Only use class and attribute names that actually exist in the
    metamodels. Don't make them up.

The step-by-step process is as follows:

1. For each relation, select the classes to be analyzed. The
    classes are always combined in pairs, in order, and contain one
     class from each metamodel.
2. For each class, select the attributes that should appear in the
     final response to meet the user's needs.
3. If the class has some container, the container class and the
    attribute that collected the class should also appear in the
    list.
3. Create the JSON array with the selected attributes for each
    metamodel.
4. Provide the final answer.

Your final answer should contain only the valid JSON and nothing
    else. Exclude any explanation or delimiter from the final
    response.

```

```
View description: {view_description}
Metamodel 1: {meta_1}
Metamodel 2: {meta_2}
List of relations: {relations}
Select elements:
"""
```

Listing A.1 – Python f-string used as prompt template in the SELECT step

```
"""
You specialize in reason on PlantUML metamodels, especially
    combining and merging them.

Given two metamodels, a list of relations containing classes'
    pairs, and a view description, your task is to define how to
    combine the given classes.

It means you must define the combination rules to combine classes
    from both metamodels.

For the input relations list, you may assume the following
    template:

{{
    "relations": [
        {{
            "name": "relationName",
            "classes": ["class_name_from_first_metamodel", "
                class_name_from_second_metamodel"]
        }}
    ]
}}

{format_instructions}

When generating the response text you should follow these rules:
Only use class and attribute names that actually exist in the
    metamodels. Don't make them up.
The combination_rule should be a string explaining how the classes
    can be connected according to the domain's semantics. It means
```

```
    explaining what kind of comparisons can be used to connect the
    classes in the relation.

The step-by-step process is as follows:

1. Select the metamodels to be analyzed for each relation in the
   list of relations.
2. For each pair of metamodel classes, analyze the domain
   considering the view description and elaborate a possible
   combination to relate both classes.
3. Create the JSON array with the combination rule for the
   relation. The combination rule is a list  that constains the
   name of the first metaclass, the combination explanation and
   the name of the second metaclass.
4. Create the JSON array with one rule per relation.
5. Provide the final answer.

Exclude any explanation or delimiter from the final response.

View description: {view_description}
Metamodel 1: {meta_1}
Metamodel 2: {meta_2}
List of relations: {relations}
Combination rules:
"""
```

Listing A.2 – Python f-string used as prompt template in the WHERE step

The prompt presented in the listing A.3 is the prompt used to run the LLM-as-judge Evaluator as explained in the Section 4.5 presented here for completeness. It is a direct adaptation of the prompt provided by LangChain as an example for evaluation [1].

```
"""
[Instruction]
Please act as an impartial judge and evaluate the quality of the
   response provided by an AI assistant to the user question
   displayed below. For this evaluation, you should primarily
   consider the following criteria:
helpfulness: How much effort would someone who knows the domain
```

1. https://docs.smith.langchain.com/evaluation/tutorials/evaluation

```
    and the VPDL languange need to make to get the prediction to
    match the reference? The less effort needed, the higher the
    score.
[Ground truth]
{{vpdl_example}}

Begin your evaluation by providing a short explanation. Be as
    objective as possible. After providing your explanation, you
    must rate the response on a scale of 1 to 10 by strictly
    following this format: "[[rating]]", for example: "Rating:
    [[5]]".

[Question]
{{view_description}}

[The Start of Assistant's Answer]
{{vpdl_draft}}

[The End of Assistant's Answer]
"""
```

Listing A.3 – Python f-string used as prompt template for the evaluator

List of Figures

List of Tables

Acronyms

AA Apprentissage Automatique. iii, iv, vii, ix, xiv, xvi

AADL Architecture Analysis and Design Language. 58

AI Artificial Intelligence. ix, xi, xvi, 3, 4, 7, 8, 10, 14, 36, 46, 48–51, 53, 54, 60, 61, 108, 111, 114, 116, 119, 120

AML AutomationML. 52, 53, 129

ANN Artificial Neural Network. 3, 7, 37, 38

AP Apprentissage Profond. iii, iv, vii–ix, xiv, xvi

API Application Programming Interface. 34, 48, 52, 76, 117

AQL Acceleo Query Language. 57

AST abstract syntax tree. 25

ATL Atlas Transformation Language. 25, 61, 67, 78, 80, 81, 83, 84, 87, 94, 115, 120, 130

BPMN Business Process Model and Notation. xiii, 12, 61

CASE Computer Aided Software Engineering. 51

CNN Convolutional Neural Network. 40, 41

CoT Chain-of-Toughts. 47, 48, 75, 116, 123

CPS Cyber-Physical-System. 2–4, 6, 8, 10, 11, 19, 20, 28, 48, 50–54, 111, 114, 119, 121

CPU Central processing unit. 39

CS Computer Science. 37

DL Deep Learning. 3, 4, 7, 8, 12–15, 17, 37, 38, 44, 51, 54, 55, 59, 60, 63, 90, 93, 96, 97, 104, 107, 111–115, 118, 119

DS Dumper System. 49, 50

DSL Domain-Specific Language. 4, 12, 15, 20, 24, 25, 31, 50, 55, 57, 58, 66, 112, 113, 118, 120

DSR Design Science Research. xiv, xvi, 14

ECL Epsilon Comparison Language. 94

EMF Eclipse Modeling Framework. 25, 32, 34, 52, 54–57, 72, 76, 101, 113, 117

EMOF Essential MOF. 25

ETL Epsilon Transformation Language. 25

EVM Event-driven Virtual Machine. 57

FML Federation Modeling Language. 56

GAN Generative Adversarial Network. 41

GAT Graph Attention Network. 45

GCN Graph Convolutional Network. 41, 45

GML Grand Modèle de Langage. viii, ix, xi, xiv, xvi, xvii

GNN Graph Neural Network. viii, ix, xi, xiv, xvi, xvii, 7, 8, 10, 14–17, 37, 40, 41, 44–46, 54, 60–64, 90, 91, 96, 98, 100, 102, 103, 107, 111, 113–115, 118–121

GPL General Purpose Language. 20

GPT Generative Pre-trained Transformer. 43, 62, 76, 87

GPU Graphics processing unit. 39

GraphSAGE Graph Sample and Aggregate. 43, 46

HGNN Heterogeneous Graph Neural Networks. 45, 46, 90–93, 96, 98, 101, 103, 107, 108, 112, 113, 115, 117–120

HMCS Heterogeneous Matching and Consistency management Suite. 58

IA Intelligence Artificielle. iii, iv, vii

IDM Ingénierie Dirigée par les Modèles. ii–ix, xi, xii, xiv, xvi, xvii, 128

IdO Internet des objets. ii

IMA Intelligent Modeling Assistant. 60

IMTA IMT Atlantique. xi, 10

IoT Internet of Things. 2

IoU Intersection over Union. 38

IP Ingénierie de Prompt. ix

IT Information Technology. xiv, 8, 14, 15, 19, 56

KG Knowledge Graph. 120

LD Langage Dédié. v, xiii, xvii

LLM Large Language Model. 8, 10, 14–17, 25, 37, 40, 41, 43, 46–48, 54, 60, 62, 63, 66, 67, 70–72, 75–77, 79, 80, 82–85, 87, 91, 111–118, 120, 121, 123, 126, 130

LM Language Model. 62

LoC Lines of Code. 103, 105

LSTM Long/Short Term Memory. 40, 41, 43, 61

MAC Multi-Agent Collaboration. 116, 117

MBE Model-Based Engineering. 19, 55, 111

MBSD Model-Based Software Development. 19

MBSE Model-Based System Engineering. 50, 51, 111

MDA™ Model-Driven Architecture. 19

MDD Model-Driven Development. 18

MDE Model-Driven Engineering. 2–5, 7, 8, 10, 11, 13–15, 17–20, 25, 26, 28–30, 38, 46, 48, 50, 54, 55, 58–61, 63, 66, 71, 87, 100, 111–115, 117, 119, 120, 128

MDSE Model-Driven Software Engineering. 18

ML Machine Learning. 3, 4, 7, 8, 13, 14, 17, 36–38, 45, 50, 51, 60–63, 66, 71, 90, 93, 96, 98, 100, 102, 107, 112–114, 116–119

MOF™ Meta-Object Facility. 24–26

MPS Meta Programming System. 25

MTBE Model Transformations by Example. 60, 61

NLP Natural Language Processing. 37, 40, 41, 60, 61

NN Neural Network. xvii, 15, 17, 20, 37–42, 44, 46, 54, 61, 98, 112, 129

OCL Object Constraint Language. 56, 61, 62

OMG™ Object Management Group. 19, 24

OSM Orthographic Software Modelling. 58

PE Prompt Engineering. 8, 47, 62, 63, 67, 71, 72, 75, 84, 87, 112, 115, 116, 123

QVT Query/View/Transformation. 77, 83

RAG Retrieval-Augmented Generation. 70, 71, 116, 120

RL Reinforcemnt Learning. 61, 121

RLHF Reinforcement Learning from Human Feedback. 47

RNA Réseau de Neurones Artificiels. iii, viii

RNN Recurrent Neural Network. viii, 7, 40, 41, 43, 61

ROC Receiver Operating Characteristic. 103–105, 129

SCP Système Cyber-Physique. ii–v, vii, ix, xi, xii

SE Software Engineering. 18, 19, 60, 62, 112, 119

SQL Structured Query Language. 31, 62

SVM Support Vector Machines. 37

SysML™ Systems Modeling Language. xiii, 12, 20, 51–53, 56, 129

TGG Triple Graph Grammars. 58

TPU Tensor processing unit. 39

UML Unified Modeling Language. xiii, 12, 22, 23, 28–30, 32–34, 53, 128, 129

URI Uniform Resource Identifier. 33, 34

VCE Volvo Construction Equipment. 48–53, 111, 129

VPDL ViewPoint Definition Language. 31–33, 53, 62, 67–69, 72, 76, 77, 79–81, 83, 84, 86, 87, 90, 91, 94–98, 100–102, 104, 107, 112, 113, 117–120, 123, 130

VQL VIATRA Query Language. 62

VSM View Specification Model. 57

XMI XML Metadata Interchange. 71

BIBLIOGRAPHY

[1] H. Zainab, A. Hesham, and M. Mahmoud, « Internet of things (IoT): definitions, challenges and recent research directions », *International Journal of Computer Applications*, vol. 128, pp. 37–47, Oct. 1, 2015, ADS Bibcode: 2015IJCA..128a..37Z. DOI: 10.5120/ijca2015906430. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2015IJCA..128a..37Z (visited on 08/05/2024).

[2] T. Sanislav and L. Miclea, « Cyber-physical systems - concept, challenges and research areas », *Journal of Control Engineering and Applied Informatics*, vol. 14, *2*, pp. 28–33, Jun. 29, 2012, Number: 2, ISSN: 1454-8658. DOI: 10.61416/ceai.v14i2.1292. [Online]. Available: http://www.ceai.srait.ro/index.php?journal=ceai&page=article&op=view&path%5B%5D=1292 (visited on 08/05/2024).

[3] S. Patel, M. Rahevar, and M. Parmar, « Data provenance and data lineage in the cloud: a survey », *International Journal of Advanced Science and Technology*, vol. 29, *5*, 2020.

[4] S. Wagenmann *et al.*, « Reference architecture for metadata management - a case study on data mining in the development of cyber-physical systems », *in 2023 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, Dec. 2023, pp. 1057–1061. DOI: 10.1109/IEEM58616.2023.10406413. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10406413 (visited on 11/06/2024).

[5] N. Jazdi, « Cyber physical systems in the context of industry 4.0 », *in 2014 IEEE International Conference on Automation, Quality and Testing, Robotics*, May 2014, pp. 1–4. DOI: 10.1109/AQTR.2014.6857843. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6857843 (visited on 08/05/2024).

[6] R. A. Nieto, *Data fusion for automated business decisions*, US Patent 7,580,878, 2009.

[7] N. Dey, A. S. Ashour, F. Shi, S. J. Fong, and J. M. R. S. Tavares, « Medical cyber-physical systems: a survey », *Journal of Medical Systems*, vol. 42, *4*, p. 74, Mar. 10, 2018, ISSN: 1573-689X. DOI: 10.1007/s10916-018-0921-x. [Online]. Available: https://doi.org/10.1007/s10916-018-0921-x (visited on 08/05/2024).

[8] S. A. Haque, S. M. Aziz, and M. Rahman, « Review of cyber-physical system in healthcare », *International Journal of Distributed Sensor Networks*, vol. 10, *4*, p. 217 415, 2014.

[9] K. Sampigethaya and R. Poovendran, « Aviation cyber–physical systems: foundations for future aircraft and air transport », *Proceedings of the IEEE*, vol. 101, *8*, pp. 1834–1855, Aug. 2013, Conference Name: Proceedings of the IEEE, ISSN: 1558-2256. DOI: 10.1109/JPROC.2012.2235131. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6480779 (visited on 08/05/2024).

[10] J. Shi, J. Wan, H. Yan, and H. Suo, « A survey of Cyber-Physical Systems », *in WCSP 2011*, Nov. 2011, pp. 1–6.

[11] P. Derler, E. A. Lee, and A. S. Vincentelli, « Modeling cyber–physical systems », *Proceedings of the IEEE*, vol. 100, *1*, pp. 13–28, 2011.

[12] B. Dafflon, N. Moalla, and Y. Ouzrout, « The challenges, approaches, and used techniques of CPS for manufacturing in industry 4.0: a literature review », *The International Journal of Advanced Manufacturing Technology*, vol. 113, *7*, pp. 2395–2412, Apr. 1, 2021, ISSN: 1433-3015. DOI: 10.1007/s00170-020-06572-4. [Online]. Available: https://doi.org/10.1007/s00170-020-06572-4 (visited on 08/05/2024).

[13] M. Brambilla, J. Cabot, and M. Wimmer, *Model-Driven Software Engineering in Practice*. Sep. 21, 2012, vol. 1, Journal Abbreviation: Synthesis Lectures on Software Engineering Publication Title: Synthesis Lectures on Software Engineering. DOI: 10.2200/S00441ED1V01Y201208SWE001.

[14] A. L. Ramos, J. V. Ferreira, and J. Barceló, « Model-based systems engineering: an emerging approach for modern systems », *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, *1*, pp. 101–111, 2011.

[15] M. A. Mohamed, G. Kardas, and M. Challenger, « Model-driven engineering tools and languages for cyber-physical systems–a systematic literature review », *IEEE Access*, vol. 9, pp. 48 605–48 630, 2021.

[16] H. Thompson *et al.*, *Platforms4CPS, Key Outcomes and Recommendations*. Steinbeis, 2018.

[17] I. Malavolta, P. Lago, H. Muccini, P. Pelliccione, and A. Tang, « What industry needs from architectural languages: a survey », *IEEE Transactions on Software Engineering*, vol. 39, *6*, pp. 869–891, 2013.

[18] S. Wolny, A. Mazak, C. Carpella, V. Geist, and M. Wimmer, « Thirteen years of sysml: a systematic mapping study », *Software and Systems Modeling*, vol. 19, *1*, pp. 111–169, 2020.

[19] A. Cicchetti, F. Ciccozzi, and A. Pierantonio, « Multi-view approaches for software and system modelling: a systematic literature review », *Software and Systems Modeling*, vol. 18, *6*, pp. 3207–3233, Dec. 1, 2019, ISSN: 1619-1374. DOI: 10.1007/s10270-018-00713-w. [Online]. Available: https://doi.org/10.1007/s10270-018-00713-w (visited on 10/08/2024).

[20] A. Bucchiarone, J. Cabot, R. F. Paige, and A. Pierantonio, « Grand challenges in model-driven engineering: an analysis of the state of the research », en, *Software and Systems Modeling*, vol. 19, *1*, pp. 5–13, Jan. 1, 2020, ISSN: 1619-1374. DOI: 10.1007/s10270-019-00773-6. [Online]. Available: https://doi.org/10.1007/s10270-019-00773-6 (visited on 07/19/2022).

[21] H. Bruneliere, E. Burger, J. Cabot, and M. Wimmer, « A feature-based survey of model view approaches », en, *Software & Systems Modeling*, vol. 18, *3*, pp. 1931–1952, Jun. 2019, ISSN: 1619-1366, 1619-1374. DOI: 10.1007/s10270-017-0622-9. [Online]. Available: http://link.springer.com/10.1007/s10270-017-0622-9 (visited on 01/26/2022).

[22] G. Rebala, A. Ravi, and S. Churiwala, « Machine learning definition and basics », *in An Introduction to Machine Learning*, G. Rebala, A. Ravi, and S. Churiwala, Eds., Cham: Springer International Publishing, 2019, pp. 1–17, ISBN: 978-3-030-15729-6. DOI: 10.1007/978-3-030-15729-6_1. [Online]. Available: https://doi.org/10.1007/978-3-030-15729-6_1 (visited on 08/05/2024).

[23] Y. Dang, Q. Lin, and P. Huang, « Aiops: real-world challenges and research innovations », *in ICSE 2019 Companion*, 2019, pp. 4–5.

[24] P. Radanliev, D. De Roure, M. Van Kleek, O. Santos, and U. Ani, « Artificial intelligence in cyber physical systems », *AI & SOCIETY*, vol. 36, *3*, pp. 783–796, Sep. 1, 2021, ISSN: 1435-5655. DOI: 10.1007/s00146-020-01049-0. [Online]. Available: https://doi.org/10.1007/s00146-020-01049-0 (visited on 08/05/2024).

[25] Y. LeCun, Y. Bengio, and G. Hinton, « Deep learning », *Nature*, vol. 521, *7553*, pp. 436–444, May 2015, Publisher: Nature Publishing Group, ISSN: 1476-4687. DOI: 10.1038/nature14539. [Online]. Available: https://www.nature.com/articles/nature14539 (visited on 09/21/2024).

[26] « What's next for AI in 2024 », MIT Technology Review. (), [Online]. Available: https://www.technologyreview.com/2024/01/04/1086046/whats-next-for-ai-in-2024/ (visited on 10/09/2024).

[27] X. Wang, Y. Zhao, and F. Pourpanah, « Recent advances in deep learning », *International Journal of Machine Learning and Cybernetics*, vol. 11, *4*, pp. 747–750, Apr. 1, 2020, ISSN: 1868-808X. DOI: 10.1007/s13042-020-01096-5. [Online]. Available: https://doi.org/10.1007/s13042-020-01096-5 (visited on 10/09/2024).

[28] W. Afzal *et al.*, « The MegaM@rt2 ECSEL project: MegaModelling at runtime – scalable model-based framework for continuous development and runtime validation of complex systems », en, *Microprocessors and Microsystems*, vol. 61, pp. 86–95, Sep. 1, 2018, ISSN: 0141-9331. DOI: 10.1016/j.micpro.2018.05.010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S014193311830022X (visited on 06/30/2023).

[29] H. Bruneliere *et al.*, « AIDOaRt: AI-augmented automation for DevOps, a model-based framework for continuous development in cyber–physical systems », en, *Microprocessors and Microsystems*, vol. 94, p. 104 672, Oct. 1, 2022, ISSN: 0141-9331. DOI: 10.1016/j.micpro.2022.104672. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0141933122002022 (visited on 06/30/2023).

[30] H. Bruneliere *et al.*, « Model-driven engineering for design-runtime interaction in complex systems: scientific challenges and roadmap », *in MDE@DeRun 2018 workshop, co-located with the Software Technologies: Applications and Foundations (STAF 2018) federation of conferences*, ser. Software Technologies: Applications and Foundations (STAF 2018) Workshops, vol. LNCS 11176, Toulouse, France, Jun. 2018. DOI: 10.1007/978-3-030-04771-9_40. [Online]. Available: https://hal.science/hal-01890878 (visited on 11/07/2024).

[31] A. Sadovykh *et al.*, « MegaM@rt2 project: mega-modelling at runtime - intermediate results and research challenges », *in Software Technology: Methods and Tools (TOOLS 2019)*, ser. Lecture Notes in Computer Science (LNCS), vol. 11771, Innopolis, Russia: Springer, Cham, Oct. 2019, pp. 393–405. DOI: 10.1007/978-3-030-29852-4_33. [Online]. Available: https://hal.science/hal-02177567 (visited on 11/07/2024).

[32] D. Bilic, E. Brosse, A. Sadovykh, D. Truscan, H. Bruneliere, and U. Ryssel, « An integrated model-based tool chain for managing variability in complex system design », *in 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, Munich, Germany: IEEE, Sep. 2019, pp. 288–293, ISBN: 978-1-72815-125-0. DOI: 10.1109/MODELS-C.2019.00045. [Online]. Available: https://ieeexplore.ieee.org/document/8904766/ (visited on 04/22/2022).

[33] G. Schneider, S. Keil, and F. Lindner, « Benefits of digitalization for business processes in semiconductor manufacturing », *in 2021 22nd IEEE International Conference on Industrial Technology (ICIT)*, vol. 1, Mar. 2021, pp. 1027–1033. DOI: 10.1109/ICIT46573.2021.9453611. [Online]. Available: https://ieeexplore.ieee.org/document/9453611/keywords#keywords (visited on 10/06/2024).

[34] O. Vermesan, *Artificial intelligence for digitising industry–applications*. CRC Press, 2022.

[35] J.-M. Favre, « Towards a basic theory to model model driven engineering », *in 3rd workshop in software model engineering, wisme*, Citeseer, 2004, pp. 262–271.

[36] F. R. Golra, A. Beugnard, F. Dagnat, S. Guerin, and C. Guychard, « Addressing modularity for heterogeneous multi-model systems using model federation », *in Companion Proceedings of the 15th International Conference on Modularity*, ser. MODULARITY Companion 2016, New York, NY, USA: Association for Computing Machinery, Mar. 14, 2016, pp. 206–211, ISBN: 978-1-4503-4033-5. DOI: 10.1145/2892664.2892701. [Online]. Available: https://doi.org/10.1145/2892664.2892701 (visited on 09/22/2024).

[37] R. F. Paige, D. S. Kolovos, L. M. Rose, N. Drivalos, and F. A. Polack, « The design of a conceptual framework and technical infrastructure for model management language engineering », *in 2009 14th IEEE International Conference on Engineering of Complex Computer Systems*, Jun. 2009, pp. 162–171. DOI: 10.1109/ICECCS.2009.14. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5090524 (visited on 09/22/2024).

[38] J.-M. Jézéquel, O. Barais, and F. Fleurey, « Model driven language engineering with kermeta », *in Generative and Transformational Techniques in Software Engineering III*, J. M. Fernandes, R. Lämmel, J. Visser, and J. Saraiva, Eds., vol. 6491, Series Title: Lecture Notes in Computer Science, Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 201–221. DOI: 10.1007/978-3-642-18023-1_5. [Online]. Available: http://link.springer.com/10.1007/978-3-642-18023-1_5 (visited on 09/22/2024).

[39] C. Hardebolle and F. Boulanger, « ModHel'x: a component-oriented approach to multi-formalism modeling », *in Models in Software Engineering*, H. Giese, Ed., vol. 5002, ISSN: 0302-9743, 1611-3349 Series Title: Lecture Notes in Computer Science, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 247–258. DOI: 10.1007/978-3-540-69073-3_26. [Online]. Available: http://link.springer.com/10.1007/978-3-540-69073-3_26 (visited on 09/22/2024).

[40] F. Vernadat, « UEML: towards a unified enterprise modelling language », *International Journal of Production Research*, vol. 40, *17*, pp. 4309–4321, Jan. 2002, ISSN: 0020-7543, 1366-588X. DOI: 10.1080/00207540210159626. [Online]. Available: http://doi.org/10.1080/00207540210159626 (visited on 09/22/2024).

[41] F. R. Golra, F. Dagnat, J. Souquières, I. Sayar, and S. Guerin, « Bridging the gap between informal requirements and formal specifications using model federation », *in Software Engineering and Formal Methods*, E. B. Johnsen and I. Schaefer, Eds., Cham: Springer International Publishing, 2018, pp. 54–69, ISBN: 978-3-319-92970-5. DOI: 10.1007/978-3-319-92970-5_4.

[42] H. Bruneliere, F. M. de Kerchove, G. Daniel, S. Madani, D. Kolovos, and J. Cabot, « Scalable model views over heterogeneous modeling technologies and resources », en, *Software and Systems Modeling*, vol. 19, *4*, pp. 827–851, Jul. 2020, ISSN: 1619-1366, 1619-1374. DOI: 10.1007/s10270-020-00794-6. [Online]. Available: http://link.springer.com/10.1007/s10270-020-00794-6 (visited on 01/11/2022).

[43] T. Mens, « A state-of-the-art survey on software merging », *IEEE Transactions on Software Engineering*, vol. 28, *5*, pp. 449–462, May 2002, Conference Name: IEEE Transactions on Software Engineering, ISSN: 1939-3520. DOI: 10.1109/TSE.2002.1000449. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1000449 (visited on 10/04/2024).

[44] O. Badreddin, T. C. Lethbridge, and A. Forward, « A novel approach to versioning and merging model and code uniformly », *in 2014 2nd International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, Jan. 2014, pp. 254–263. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7018472 (visited on 10/04/2024).

[45] J. Favre, « Megamodelling and etymology-a story of words: from MED to MDE via MODEL in five millenium, s dagstuhl seminar 05161 on" transformation techniques in software engineering », *Dagsthul, Allemagne*, 2005.

[46] C. Guychard, S. Guerin, A. Koudri, A. Beugnard, and F. Dagnat, « Conceptual interoperability through models federation », *in Semantic information federation community workshop*, vol. 23, 2013.

[47] F. Jouault, B. Vanhooff, H. Bruneliere, G. Doux, Y. Berbers, and J. Bezivin, « Inter-DSL coordination support by combining megamodeling and model weaving », *in Proceedings of the 2010 ACM Symposium on Applied Computing*, ser. SAC '10, New York, NY, USA: Association for Computing Machinery, Mar. 22, 2010, pp. 2011–2018, ISBN: 978-1-60558-

639-7. DOI: 10.1145/1774088.1774511. [Online]. Available: https://dl.acm.org/doi/10.1145/1774088.1774511 (visited on 10/04/2024).

[48] J. Pietron, A. Raschke, J. Exelmans, and M. Tichy, « Collaboration and versioning framework–a systematic top-down approach », *in 2023 ACM/IEEE international conference on model driven engineering languages and systems companion (MODELS-c)*, IEEE, 2023, pp. 767–777.

[49] R. Manellanga and I. David, « Participatory and collaborative modeling of sustainable systems: a systematic review », *in 2023 ACM/IEEE international conference on model driven engineering languages and systems companion (MODELS-c)*, IEEE, 2024.

[50] S. Bennani, S. Ebersold, M. El Hamlaoui, B. Coulette, and M. Nassar, « A collaborative decision approach for alignment of heterogeneous models », *in 2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, ISSN: 2641-8169, Jun. 2019, pp. 112–117. DOI: 10.1109/WETICE.2019.00032. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8795440 (visited on 10/04/2024).

[51] M. D. Del Fabro, J. Bézivin, F. Jouault, P. Valduriez, *et al.*, « Applying generic model management to data mapping. », *in Base de données avancées (BDA 2005)*, Place: Saint-Malo, France, Oct. 20, 2005. [Online]. Available: https://pubs.dbs.uni-leipzig.de/se/files/Fabro2005ApplyingGenericModel.pdf.

[52] M. D. Del Fabro, J. Bézivin, and P. Valduriez, « Weaving models with the eclipse AMW plugin », *in Eclipse modeling symposium, eclipse summit europe*, vol. 2006, Citeseer, 2006, pp. 37–44.

[53] E. Felix, D. Lopes, and O. S. Jr., « A framework based on model driven engineering and model weaving to support data-driven interoperability for smart grid applications », *in Proceedings of the 2020 European Symposium on Software Engineering*, ser. ESSE '20, New York, NY, USA: Association for Computing Machinery, Dec. 21, 2020, pp. 30–36, ISBN: 978-1-4503-7762-1. DOI: 10.1145/3393822.3432341. [Online]. Available: https://dl.acm.org/doi/10.1145/3393822.3432341 (visited on 10/04/2024).

[54] A. Colantoni, L. Berardinelli, and M. Wimmer, « DevOpsML: towards modeling DevOps processes and platforms », *in Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, ser. MODELS '20, New York, NY, USA: Association for Computing Machinery, Oct. 26, 2020, pp. 1–10, ISBN: 978-1-4503-8135-2. DOI: 10.1145/3417990.3420203. [Online]. Available: https://dl.acm.org/doi/10.1145/3417990.3420203 (visited on 10/04/2024).

[55] C. Boudjemila, F. Dagnat, and S. Martínez, « Maintaining security consistency during system development with security-oriented model federation », *in Proceedings of the 2024 International Conference on Software and Systems Processes*, ser. ICSSP '24, New York, NY, USA: Association for Computing Machinery, Sep. 4, 2024, pp. 66–76, ISBN: 9798400709913. DOI: 10.1145/3666015.3666016. [Online]. Available: https://dl.acm.org/doi/10.1145/3666015.3666016 (visited on 10/06/2024).

[56] International Organization for Standardization, « Automation systems and integration - architecture, communications and integration frameworks », manual, 2023. [Online]. Available: https://www.iso.org/committee/54192.html.

[57] T. Goldschmidt, S. Becker, and E. Burger, « Towards a tool-oriented taxonomy of view-based modelling », presented at the Modellierung 2012, Gesellschaft für Informatik e.V., 2012, pp. 59–74, ISBN: 978-3-88579-295-6. [Online]. Available: https://dl.gi.de/handle/20.500.12116/18148 (visited on 10/07/2024).

[58] International Organization for Standardization, « Systems and software engineering — architecture description », manual, 2022. [Online]. Available: https://www.iso.org/standard/74393.html.

[59] A. A. Shah, A. A. Kerzhner, D. Schaefer, and C. J. J. Paredis, « Multi-view modeling to support embedded systems engineering in SysML », *in Graph Transformations and Model-Driven Engineering: Essays Dedicated to Manfred Nagl on the Occasion of his 65th Birthday*, G. Engels, C. Lewerentz, W. Schäfer, A. Schürr, and B. Westfechtel, Eds., Berlin, Heidelberg: Springer, 2010, pp. 580–601, ISBN: 978-3-642-17322-6. DOI: 10.1007/978-3-642-17322-6_25. [Online]. Available: https://doi.org/10.1007/978-3-642-17322-6_25 (visited on 10/08/2024).

[60] A. Cicchetti, F. Ciccozzi, and T. Leveque, « Supporting incremental synchronization in hybrid multi-view modelling », *in Models in Software Engineering*, J. Kienzle, Ed., Berlin, Heidelberg: Springer, 2012, pp. 89–103, ISBN: 978-3-642-29645-1. DOI: 10.1007/978-3-642-29645-1_11.

[61] J.-C. Bach, A. Beugnard, J. Champeau, F. Dagnat, S. Guérin, and S. Martínez, « 10 years of model federation with openflexo: challenges and lessons learned », *in Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems*, ser. MODELS '24, New York, NY, USA: Association for Computing Machinery, Sep. 22, 2024, pp. 25–36, ISBN: 9798400705045. DOI: 10.1145/3640310.3674084. [Online]. Available: https://dl.acm.org/doi/10.1145/3640310.3674084 (visited on 10/07/2024).

[62] G. Csertan, G. Huszerl, I. Majzik, Z. Pap, A. Pataricza, and D. Varro, « VIATRA - visual automated transformations for formal verification and validation of UML models », en, *in Proceedings 17th IEEE International Conference on Automated Software Engineering,*, ISSN: 1938-4300, Edinburgh, UK: IEEE, Sep. 2002, pp. 267–270, ISBN: 978-0-7695-1736-0. DOI: 10.1109/ASE.2002.1115027. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1115027 (visited on 10/09/2024).

[63] G. Bergmann *et al.*, « Viatra 3: a reactive model transformation platform », *in Theory and Practice of Model Transformations*, D. Kolovos and M. Wimmer, Eds., vol. 9152, Series Title: Lecture Notes in Computer Science, Cham: Springer International Publishing, 2015, pp. 101–110. DOI: 10.1007/978-3-319-21155-8_8. [Online]. Available: https://link.springer.com/10.1007/978-3-319-21155-8_8 (visited on 10/08/2024).

[64] H. Bruneliere, J. G. Perez, M. Wimmer, and J. Cabot, « EMF views: a view mechanism for integrating heterogeneous models », en, presented at the 34th International Conference on Conceptual Modeling (ER 2015), Oct. 19, 2015. DOI: 10.1007/978-3-319-25264-3_23. [Online]. Available: https://hal.inria.fr/hal-01159205 (visited on 01/07/2022).

[65] R. Eramo *et al.*, « Model-driven design-runtime interaction in safety critical system development: an experience report. », en, *The Journal of Object Technology*, vol. 18, *2*, 1:1, 2019, ISSN: 1660-1769. DOI: 10.5381/jot.2019.18.2.a1. [Online]. Available: http://www.jot.fm/contents/issue_2019_02/article1.html (visited on 01/13/2022).

[66] J. Cederbladh *et al.*, « Towards automating model-based systems engineering in industry - an experience report », *in 2024 IEEE International Systems Conference (SysCon)*, ISSN: 2472-9647, Montreal, Canada, Apr. 2024, pp. 1–8. DOI: 10.1109/SysCon61195.2024.10553610. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10553610 (visited on 08/01/2024).

[67] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th edition. Hoboken: Pearson, May 8, 2020, 1136 pp., ISBN: 978-0-13-461099-3.

[68] S. Shafiq, A. Mashkoor, C. Mayr-Dorn, and A. Egyed, « A literature review of using machine learning in software development life cycle stages », *IEEE Access*, vol. 9, pp. 140 896–140 920, 2021, Conference Name: IEEE Access, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3119746.

[69] L. Iovino, A. Barriga Rodriguez, A. Rutle, and R. Heldal, « Model repair with quality-based reinforcement learning », eng, *19*, 2020, Accepted: 2021-04-12T07:41:21Z Publisher: AITO — Association Internationale pour les Technologies Objets, ISSN: 1660-1769. DOI:

10.5381/JOT.2020.19.2.A17. [Online]. Available: https://hvlopen.brage.unit.no/hvlopen-xmlui/handle/11250/2737208 (visited on 07/19/2022).

[70] P. T. Nguyen, J. Di Rocco, D. Di Ruscio, A. Pierantonio, and L. Iovino, « Automated classification of metamodel repositories: a machine learning approach », *in 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS)*, Sep. 2019, pp. 272–282. DOI: 10.1109/MODELS.2019.00011.

[71] K. Lano, S. Yassipour-Tehrani, and M. A. Umar, « Automated requirements formalisation for agile MDE », *in 2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, Oct. 2021, pp. 173–180. DOI: 10.1109/MODELS-C53483.2021.00030.

[72] A. Barriga, A. Rutle, and R. Heldal, « AI-powered model repair: an experience report—lessons learned, challenges, and opportunities », en, *Software and Systems Modeling*, Feb. 22, 2022, ISSN: 1619-1366, 1619-1374. DOI: 10.1007/s10270-022-00983-5. [Online]. Available: https://link.springer.com/10.1007/s10270-022-00983-5 (visited on 03/15/2022).

[73] N. Nassar, H. Radke, and T. Arendt, « Rule-based repair of EMF models: an automated interactive approach », en, *in Theory and Practice of Model Transformation*, E. Guerra and M. van den Brand, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2017, pp. 171–181, ISBN: 978-3-319-61473-1. DOI: 10.1007/978-3-319-61473-1_12.

[74] R. Kretschmer, D. E. Khelladi, and A. Egyed, « An automated and instant discovery of concrete repairs for model inconsistencies », *in Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings*, ser. ICSE '18, New York, NY, USA: Association for Computing Machinery, May 27, 2018, pp. 298–299, ISBN: 978-1-4503-5663-3. DOI: 10.1145/3183440.3194979. [Online]. Available: https://doi.org/10.1145/3183440.3194979 (visited on 07/19/2022).

[75] L. Burgueno, J. Cabot, and S. Gerard, « An LSTM-based neural network architecture for model transformations », *in 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS)*, IEEE, Sep. 2019, pp. 294–299, ISBN: 978-1-72812-536-7. DOI: 10.1109/MODELS.2019.00013. [Online]. Available: https://ieeexplore.ieee.org/document/8906971/.

[76] D. E. Khelladi, R. Kretschmer, and A. Egyed, « Detecting and exploring side effects when repairing model inconsistencies », *in Proceedings of the 12th ACM SIGPLAN international conference on software language engineering*, 2019, pp. 113–126.

[77] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, « The Graph Neural Network Model », *IEEE TRANSACTIONS ON NEURAL NETWORKS*, vol. 20, *1*, p. 20, Jan. 2009, 01896, ISSN: 1941-0093. DOI: 10.1109/TNN.2008.2005605.

[78] R. Clarisó and J. Cabot, « Applying graph kernels to model-driven engineering problems », en, *in Proceedings of the 1st International Workshop on Machine Learning and Software Engineering in Symbiosis*, ser. MASES 2018, New York, NY, USA: ACM, Sep. 2018, pp. 1–5, ISBN: 978-1-4503-5972-6. DOI: 10.1145/3243127.3243128. [Online]. Available: https://doi.org/10.1145/3243127.3243128 (visited on 06/16/2022).

[79] J. A. H. López and J. S. Cuadrado, « Generating structurally realistic models with deep autoregressive networks », *IEEE Transactions on Software Engineering*, vol. 49, *4*, pp. 2661–2676, Apr. 2023, Conference Name: IEEE Transactions on Software Engineering, ISSN: 1939-3520. DOI: 10.1109/TSE.2022.3228630.

[80] J. Di Rocco, C. Di Sipio, D. Di Ruscio, and P. T. Nguyen, « A GNN-based recommender system to assist the specification of metamodels and models », *in 2021 ACM/IEEE 24th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, Oct. 2021, pp. 70–81. DOI: 10.1109/MODELS50736.2021.00016.

[81] M. Weyssow, H. Sahraoui, and E. Syriani, « Recommending metamodel concepts during modeling activities with pre-trained language models », en, *Software and Systems Modeling*, vol. 21, *3*, pp. 1071–1089, Jun. 2022, arXiv:2104.01642 [cs], ISSN: 1619-1366, 1619-1374. DOI: 10.1007/s10270-022-00975-5. [Online]. Available: http://arxiv.org/abs/2104.01642 (visited on 07/19/2022).

[82] J. A. Hernández López, C. Durá, and J. S. Cuadrado, « Word embeddings for model-driven engineering », *in 2023 ACM/IEEE 26th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, Oct. 2023, pp. 151–161. DOI: 10.1109/MODELS58315.2023.00036. [Online]. Available: https://ieeexplore.ieee.org/document/10344175?denied= (visited on 01/28/2024).

[83] M. B. Chaaben, L. Burgueño, and H. Sahraoui, « Towards using few-shot prompt learning for automating model completion », *in 2023 IEEE/ACM 45th International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*, ISSN: 2832-7632, May 2023, pp. 7–12. DOI: 10.1109/ICSE-NIER58687.2023.00008. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10173909/authors#authors (visited on 12/16/2023).

[84] J. Cámara, J. Troya, L. Burgueño, and A. Vallecillo, « On the assessment of generative AI in modeling tasks: an experience report with ChatGPT and UML », en, *Software and Systems Modeling*, vol. 22, *3*, pp. 781–793, Jun. 1, 2023, ISSN: 1619-1374. DOI: 10.1007/s10270-023-01105-5. [Online]. Available: https://doi.org/10.1007/s10270-023-01105-5 (visited on 11/30/2023).

[85] K. Chen, Y. Yang, B. Chen, J. A. Hernández López, G. Mussbacher, and D. Varró, « Automated domain modeling with large language models: a comparative study », en, Jul. 1, 2023, Publisher: Zenodo Version Number: v5. DOI: 10.5281/ZENODO.8118642. [Online]. Available: https://zenodo.org/record/8118642 (visited on 11/30/2023).

[86] Object Management Group, « OMG unified modeling language (OMG UML). version 2.5.1 », Object Management Group (OMG), manual, Dec. 2017. [Online]. Available: https://www.omg.org/spec/UML/2.5.1.

[87] Object Management Group. « Business process model and notation (BPMN™) version 2.0.2 ». (Jan. 2014), [Online]. Available: http://www.omg.org/spec/BPMN/2.0.2/ (visited on 09/25/2024).

[88] Object Management Group. « OMG systems modeling language (OMG SysML™) version 1.6 ». (Nov. 2019), [Online]. Available: https://www.omg.org/spec/SysML/1.6/ (visited on 09/25/2024).

[89] A. R. Hevner, S. T. March, J. Park, and S. Ram, « Design science in information systems research », *MIS Quarterly*, vol. 28, *1*, pp. 75–105, 2004, Publisher: Management Information Systems Research Center, University of Minnesota, ISSN: 0276-7783. DOI: 10.2307/25148625. [Online]. Available: https://www.jstor.org/stable/25148625 (visited on 07/31/2024).

[90] J. Venable, J. Pries-Heje, and R. Baskerville, « FEDS: a framework for evaluation in design science research », *European Journal of Information Systems*, vol. 25, *1*, pp. 77–89, Jan. 1, 2016, Publisher: Taylor & Francis _eprint: https://doi.org/10.1057/ejis.2014.36, ISSN: 0960-085X. DOI: 10.1057/ejis.2014.36. [Online]. Available: https://doi.org/10.1057/ejis.2014.36 (visited on 08/01/2024).

[91] M. Sharbaf, B. Zamani, and G. Sunyé, « CoMPers: a configurable conflict management framework for personalized collaborative modeling », *Journal of Systems and Software*, vol. 219, p. 112 227, Jan. 1, 2025, ISSN: 0164-1212. DOI: 10.1016/j.jss.2024.112227. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121224002711 (visited on 11/22/2024).

[92] G. Sunyé, « Model consistency for distributed collaborative modeling », *in Modelling Foundations and Applications*, A. Anjorin and H. Espinoza, Eds., Cham: Springer International Publishing, 2017, pp. 197–212, ISBN: 978-3-319-61482-3. DOI: 10.1007/978-3-319-61482-3_12.

[93] M. Sharbaf, B. Zamani, and G. Sunyé, « Automatic resolution of model merging conflicts using quality-based reinforcement learning », *Journal of Computer Languages*, vol. 71, p. 101 123, Aug. 1, 2022, ISSN: 2590-1184. DOI: 10.1016/j.cola.2022.101123. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2590118422000260 (visited on 11/22/2024).

[94] *Model, n. & adj. In Oxford English Dictionary*, 3rd ed., Oxford University Press, Mar. 2, 2023. DOI: 10.1093/OED/3984201854. [Online]. Available: https://oed.com/dictionary/model_n (visited on 09/23/2024).

[95] W. Hodges, *A Shorter Model Theory*. Cambridge University Press, Apr. 10, 1997, 322 pp., Google-Books-ID: S6QYeuo4p1EC, ISBN: 978-0-521-58713-6.

[96] K. Börner, K. W. Boyack, S. Milojević, and S. Morris, « An introduction to modeling science: basic model types, key definitions, and a general framework for the comparison of process models », *in Models of Science Dynamics*, A. Scharnhorst, K. Börner, and P. Van Den Besselaar, Eds., Series Title: Understanding Complex Systems, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 3–22. DOI: 10.1007/978-3-642-23068-4_1. [Online]. Available: https://link.springer.com/10.1007/978-3-642-23068-4_1 (visited on 11/08/2024).

[97] S. Mellor, A. Clark, and T. Futagami, « Model-driven development - guest editor's introduction », *IEEE Software*, vol. 20, *5*, pp. 14–18, Sep. 2003, ISSN: 0740-7459. DOI: 10.1109/MS.2003.1231145. [Online]. Available: http://ieeexplore.ieee.org/document/1231145/ (visited on 09/24/2024).

[98] S. Kent, « Model driven engineering », *in Integrated Formal Methods*, M. Butler, L. Petre, and K. Sere, Eds., Berlin, Heidelberg: Springer, 2002, pp. 286–298, ISBN: 978-3-540-47884-3. DOI: 10.1007/3-540-47884-1_16.

[99] B. Selic, « The pragmatics of model-driven development », *IEEE Software*, vol. 20, *5*, pp. 19–25, Sep. 2003, Conference Name: IEEE Software, ISSN: 1937-4194. DOI: 10.1109/MS.2003.1231146. [Online]. Available: https://ieeexplore.ieee.org/document/1231146/?arnumber=1231146&tag=1 (visited on 11/08/2024).

[100] E. Seidewitz, « What models mean », *IEEE Software*, vol. 20, *5*, pp. 26–32, Sep. 2003, Conference Name: IEEE Software, ISSN: 1937-4194. DOI: 10.1109/MS.2003.1231147. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1231147 (visited on 11/08/2024).

[101] B. Hailpern and P. Tarr, « Model-driven development: the good, the bad, and the ugly », *IBM Systems Journal*, vol. 45, *3*, pp. 451–461, 2006, ISSN: 0018-8670. DOI: 10.1147/sj.453.0451. [Online]. Available: http://ieeexplore.ieee.org/document/5386628/ (visited on 11/08/2024).

[102] O. Pastor, S. España, J. I. Panach, and N. Aquino, « Model-driven development », *Informatik-Spektrum*, vol. 31, *5*, pp. 394–407, Oct. 1, 2008, ISSN: 1432-122X. DOI: 10.1007/s00287-008-0275-8. [Online]. Available: https://doi.org/10.1007/s00287-008-0275-8 (visited on 09/24/2024).

[103] H. Bruneliere, J. Cabot, F. Jouault, and F. Madiot, « MoDisco: a generic and extensible framework for model driven reverse engineering », *in Proceedings of the 25th IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE '10, New York, NY, USA: Association for Computing Machinery, Sep. 20, 2010, pp. 173–174, ISBN: 978-1-4503-0116-9. DOI: 10.1145/1858996.1859032. [Online]. Available: https://dl.acm.org/doi/10.1145/1858996.1859032 (visited on 09/24/2024).

[104] S. Assar, « Model driven requirements engineering: mapping the field and beyond », *in 2014 IEEE 4th International Model-Driven Requirements Engineering Workshop (MoDRE)*, Aug. 2014, pp. 1–6. DOI: 10.1109/MoDRE.2014.6890820. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6890820 (visited on 09/24/2024).

[105] P. Mohagheghi, V. Dehlen, and T. Neple, « Definitions and approaches to model quality in model-based software development – a review of literature », *Information and Software Technology*, Quality of UML Models, vol. 51, *12*, pp. 1646–1669, Dec. 1, 2009, ISSN: 0950-5849. DOI: 10.1016/j.infsof.2009.04.004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584909000457 (visited on 09/25/2024).

[106] Object Management Group. « MDA guide version 1.0.1 ». (2003), [Online]. Available: https://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf.

[107] J. Davies, J. Gibbons, J. Welch, and E. Crichton, « Model-driven engineering of information systems: 10 years and 1000 versions », *Science of Computer Programming*, Special issue on Success Stories in Model Driven Engineering, vol. 89, pp. 88–104, Sep. 1, 2014, ISSN: 0167-6423. DOI: 10.1016/j.scico.2013.02.002. [Online]. Available: https:

//www.sciencedirect.com/science/article/pii/S0167642313000270 (visited on 09/25/2024).

[108] J. Hutchinson, J. Whittle, and M. Rouncefield, « Model-driven engineering practices in industry: social, organizational and managerial factors that lead to success or failure », *Science of Computer Programming*, Special issue on Success Stories in Model Driven Engineering, vol. 89, pp. 144–161, Sep. 1, 2014, ISSN: 0167-6423. DOI: 10.1016/j.scico.2013.03.017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167642313000786 (visited on 09/25/2024).

[109] J.-M. Bruel, B. Combemale, I. Ober, and H. Raynal, « MDE in practice for computational science », *Procedia Computer Science*, International Conference On Computational Science, ICCS 2015, vol. 51, pp. 660–669, Jan. 1, 2015, ISSN: 1877-0509. DOI: 10.1016/j.procs.2015.05.182. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050915009904 (visited on 10/04/2024).

[110] T. Nepomuceno, T. Carneiro, P. H. Maia, M. Adnan, T. Nepomuceno, and A. Martin, « AutoIoT: a framework based on user-driven MDE for generating IoT applications », *in Proceedings of the 35th Annual ACM Symposium on Applied Computing*, ser. SAC '20, New York, NY, USA: Association for Computing Machinery, Mar. 30, 2020, pp. 719–728, ISBN: 978-1-4503-6866-7. DOI: 10.1145/3341105.3373873. [Online]. Available: https://dl.acm.org/doi/10.1145/3341105.3373873 (visited on 10/04/2024).

[111] H. Sartaj, S. Ali, T. Yue, and K. Moberg, « Model-based digital twins of medicine dispensers for healthcare IoT applications », *Software: Practice and Experience*, vol. 54, *6*, pp. 1172–1192, 2024, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.3311, ISSN: 1097-024X. DOI: 10.1002/spe.3311. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.3311 (visited on 10/04/2024).

[112] N. Sánchez-Gómez, J. Torres-Valderrama, J. A. García-García, J. J. Gutiérrez, and M. J. Escalona, « Model-based software design and testing in blockchain smart contracts: a systematic literature review », *IEEE Access*, vol. 8, pp. 164 556–164 569, 2020, Conference Name: IEEE Access, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3021502. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9186040 (visited on 09/25/2024).

[113] E. Jeong, D. Jeong, and S. Ha, « Dataflow model–based software synthesis framework for parallel and distributed embedded systems », *ACM Trans. Des. Autom. Electron. Syst.*, vol. 26, *5*, 35:1–35:38, Jun. 5, 2021, ISSN: 1084-4309. DOI: 10.1145/3447680. [Online]. Available: https://dl.acm.org/doi/10.1145/3447680 (visited on 09/25/2024).

[114] F. Büttner *et al.*, « Model-driven standardization of public authority data interchange », *Science of Computer Programming*, vol. 89, pp. 162–175, Sep. 2014, ISSN: 01676423. DOI: 10.1016/j.scico.2013.03.009. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167642313000695 (visited on 09/25/2024).

[115] J. Parri, S. Sampietro, and E. Vicario, « FaultFlow: a tool supporting an MDE approach for timed failure logic analysis », *in 2021 17th European Dependable Computing Conference (EDCC)*, ISSN: 2641-810X, Sep. 2021, pp. 25–32. DOI: 10.1109/EDCC53658.2021.00011. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9603569 (visited on 10/02/2024).

[116] L. Favre, « A framework for modernizing non-mobile software: a model-driven engineering approach », *in Research Anthology on Recent Trends, Tools, and Implications of Computer Programming*, IGI Global, 2021, pp. 320–345, ISBN: 978-1-79983-016-0. DOI: 10.4018/978-1-7998-3016-0.ch015. [Online]. Available: https://www.igi-global.com/chapter/a-framework-for-modernizing-non-mobile-software/www.igi-global.com/chapter/a-framework-for-modernizing-non-mobile-software/261033 (visited on 10/02/2024).

[117] V. Cortellessa, R. Eramo, and M. Tucci, « From software architecture to analysis models and back: model-driven refactoring aimed at availability improvement », *Information and Software Technology*, vol. 127, p. 106 362, Nov. 1, 2020, ISSN: 0950-5849. DOI: 10.1016/j.infsof.2020.106362. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584920301300 (visited on 10/02/2024).

[118] F. Bordeleau, B. Combemale, R. Eramo, M. van den Brand, and M. Wimmer, « Towards model-driven digital twin engineering: current opportunities and future challenges », *in Systems Modelling and Management*, Ö. Babur, J. Denil, and B. Vogel-Heuser, Eds., Cham: Springer International Publishing, 2020, pp. 43–54, ISBN: 978-3-030-58167-1. DOI: 10.1007/978-3-030-58167-1_4.

[119] G. N. Schroeder, C. Steinmetz, R. N. Rodrigues, R. V. B. Henriques, A. Rettberg, and C. E. Pereira, « A methodology for digital twin modeling and deployment for industry 4.0 », *Proceedings of the IEEE*, vol. 109, *4*, pp. 556–567, Apr. 2021, Conference Name: Proceedings of the IEEE, ISSN: 1558-2256. DOI: 10.1109/JPROC.2020.3032444. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9247401 (visited on 10/02/2024).

[120] M. Heithoff, A. Hellwig, J. Michael, and B. Rumpe, « Digital twins for sustainable software systems », *in 2023 IEEE/ACM 7th International Workshop on Green And Sustainable Software (GREENS)*, May 2023, pp. 19–23. DOI: 10.1109/GREENS59328.2023.

00010. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10190835 (visited on 10/02/2024).

[121] Merriam-Webster. « System ». (2024), [Online]. Available: https://www.merriam-webster.com/dictionary/system.

[122] M. Fowler, *Domain-specific languages*. Pearson Education, 2010.

[123] A. Bucchiarone, S. Martella, H. Muccini, and M. Fusco, *Dsl4gar: a domain specific language for gamification rules definition, simulation and deployment*, Rochester, NY, Feb. 14, 2023. DOI: 10.2139/ssrn.4358088. [Online]. Available: https://papers.ssrn.com/abstract=4358088 (visited on 11/11/2024).

[124] B. Jahić, N. Guelfi, and B. Ries, « SEMKIS-DSL: a domain-specific language to support requirements engineering of datasets and neural network recognition », *Information*, vol. 14, *4*, p. 213, Apr. 2023, Number: 4 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2078-2489. DOI: 10.3390/info14040213. [Online]. Available: https://www.mdpi.com/2078-2489/14/4/213 (visited on 11/11/2024).

[125] A. Iung *et al.*, « Systematic mapping study on domain-specific language development tools », *Empirical Software Engineering*, vol. 25, *5*, pp. 4205–4249, Sep. 1, 2020, ISSN: 1573-7616. DOI: 10.1007/s10664-020-09872-1. [Online]. Available: https://doi.org/10.1007/s10664-020-09872-1 (visited on 09/25/2024).

[126] J. Bézivin and O. Gerbé, « Towards a precise definition of the OMG/MDA framework », *in Proceedings 16th annual international conference on automated software engineering (ASE 2001)*, IEEE, 2001, pp. 273–280.

[127] N. Guarino, C. Welty, *et al.*, « Towards a methodology for ontology-based model engineering », *in Proceedings of the ECOOP-2000 workshop on model engineering*, 2000.

[128] J. Warmer and A. Kleppe, *The object constraint language: precise modeling with UML*. Addison-Wesley Longman Publishing Co., Inc., 1998.

[129] M. Voelter, *DSL Engineering: Designing, Implementing and Using Domain-Specific Languages*. S.l.: CreateSpace Independent Publishing Platform, Jan. 23, 2013, 558 pp., ISBN: 978-1-4812-1858-0.

[130] J. Cabot, R. Clarisó, and D. Riera, « UMLtoCSP: a tool for the formal verification of UML/OCL models using constraint programming », *in Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE '07, New York, NY, USA: Association for Computing Machinery, Nov. 5, 2007, pp. 547–548, ISBN: 978-1-59593-882-4. DOI: 10.1145/1321631.1321737. [Online]. Available: https://dl.acm.org/doi/10.1145/1321631.1321737 (visited on 10/02/2024).

[131] K. Jin and K. Lano, « OCL-based test case prioritisation using AgileUML », *in Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, ser. MODELS '22, New York, NY, USA: Association for Computing Machinery, Nov. 9, 2022, pp. 607–611, ISBN: 978-1-4503-9467-3. DOI: 10.1145/3550356.3561593. [Online]. Available: https://dl.acm.org/doi/10.1145/3550356.3561593 (visited on 10/02/2024).

[132] J. S. Cuadrado, E. Guerra, and J. de Lara, « Static analysis of model transformations », *IEEE Transactions on Software Engineering*, vol. 43, *9*, pp. 868–897, Sep. 2017, Conference Name: IEEE Transactions on Software Engineering, ISSN: 1939-3520. DOI: 10.1109/TSE.2016.2635137. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7765073 (visited on 10/02/2024).

[133] C. A. González and J. Cabot, « Formal verification of static software models in MDE: a systematic review », *Information and Software Technology*, vol. 56, *8*, pp. 821–838, Aug. 1, 2014, ISSN: 0950-5849. DOI: 10.1016/j.infsof.2014.03.003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584914000627 (visited on 10/02/2024).

[134] F. Jouault, F. Allilaire, J. Bézivin, and I. Kurtev, « ATL: a model transformation tool », *Science of Computer Programming*, Special Issue on Second issue of experimental software and toolkits (EST), vol. 72, *1*, pp. 31–39, Jun. 1, 2008, ISSN: 0167-6423. DOI: 10.1016/j.scico.2007.08.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167642308000439 (visited on 10/15/2024).

[135] D. S. Kolovos, R. F. Paige, and F. A. Polack, « Eclipse development tools for epsilon », *in Eclipse summit Europe, eclipse modeling symposium*, vol. 20062, 2006, p. 200.

[136] A. Jossic, M. D. del Fabro, J.-P. Lerat, J. Bezivin, and F. Jouault, « Model integration with model weaving: a case study in system architecture », *in 2007 International Conference on Systems Engineering and Modeling*, Mar. 2007, pp. 79–84. DOI: 10.1109/ICSEM.2007.373336. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/4243721 (visited on 10/15/2024).

[137] C. Clasen, F. Jouault, and J. Cabot, « VirtualEMF: a model virtualization tool », *in Advances in conceptual modeling. Recent developments and new directions: ER 2011 workshops FP-UML, MoRE-BI, onto-CoM, SeCoGIS, variability@ ER, WISM, brussels, belgium, october 31-november 3, 2011. Proceedings 30*, Springer, 2011, pp. 332–335.

[138] H. Bruneliere, « Generic model-based approaches for software reverse engineering and comprehension », Ph.D. dissertation, Université de Nantes, 2018.

[139] X. Su, X. Yan, and C.-L. Tsai, « Linear regression », *WIREs Computational Statistics*, vol. 4, *3*, pp. 275–294, 2012, _eprint: https://doi.org/10.1002/wics.1198, ISSN: 1939-0068. DOI: 10.1002/wics.1198. [Online]. Available: https://doi.org/10.1002/wics.1198 (visited on 08/30/2024).

[140] S. Suthaharan, « Decision tree learning », *in Machine Learning Models and Algorithms for Big Data Classification: Thinking with Examples for Effective Learning*, S. Suthaharan, Ed., Boston, MA: Springer US, 2016, pp. 237–269, ISBN: 978-1-4899-7641-3. DOI: 10.1007/978-1-4899-7641-3_10. [Online]. Available: https://doi.org/10.1007/978-1-4899-7641-3_10 (visited on 08/30/2024).

[141] M. A. Chandra and S. S. Bedi, « Survey on SVM and their application in imageclassification », *International Journal of Information Technology*, vol. 13, *5*, pp. 1–11, Oct. 1, 2021, ISSN: 2511-2112. DOI: 10.1007/s41870-017-0080-1. [Online]. Available: https://doi.org/10.1007/s41870-017-0080-1 (visited on 08/30/2024).

[142] S. Russell and P. Norvig, « The foundations of artificial intelligence (introduction) », *in Artificial Intelligence: A Modern Approach*, 4th edition, Hoboken: Pearson, May 8, 2020, pp. 5–27, ISBN: 978-0-13-461099-3.

[143] T. M. Mitchell, *Machine Learning*, 1st edition. New York: McGraw-Hill Education, Mar. 1, 1997, 432 pp., ISBN: 978-0-07-042807-2.

[144] S. J. Pan and Q. Yang, « A survey on transfer learning », *IEEE Transactions on knowledge and data engineering*, vol. 22, *10*, pp. 1345–1359, 2009, Publisher: IEEE.

[145] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, « Object detection in 20 years: a survey », *Proceedings of the IEEE*, vol. 111, *3*, pp. 257–276, Mar. 2023, Conference Name: Proceedings of the IEEE, ISSN: 1558-2256. DOI: 10.1109/JPROC.2023.3238524. [Online]. Available: https://ieeexplore.ieee.org/document/10028728 (visited on 10/21/2024).

[146] V. Nair and G. E. Hinton, « Rectified linear units improve restricted boltzmann machines », *in Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[147] D. P. Kingma and J. Ba, « Adam: a method for stochastic optimization », *arXiv e-prints*, arXiv–1412, 2014.

[148] S. Ioffe, « Batch renormalization: towards reducing minibatch dependence in batch-normalized models », *Advances in neural information processing systems*, vol. 30, 2017.

[149] D. E. Rumelhart, R. Durbin, R. Golden, and Y. Chauvin, « Backpropagation: the basic theory », *in Backpropagation*, Psychology Press, 2013, pp. 1–34.

[150] I. Bello, B. Zoph, V. Vasudevan, and Q. V. Le, « Neural optimizer search with reinforcement learning », *in International conference on machine learning*, PMLR, 2017, pp. 459–468.

[151] N. P. Jouppi *et al.*, « In-datacenter performance analysis of a tensor processing unit », *in Proceedings of the 44th annual international symposium on computer architecture*, 2017, pp. 1–12.

[152] A. Krizhevsky, I. Sutskever, and G. E. Hinton, « Imagenet classification with deep convolutional neural networks », *Advances in neural information processing systems*, vol. 25, 2012.

[153] S. Hochreiter, « Long short-term memory », *Neural Computation MIT-Press*, 1997.

[154] A. Vaswani *et al.*, « Attention is all you need », *in Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017.

[155] F. A. Gers, J. Schmidhuber, and F. Cummins, « Learning to forget: continual prediction with LSTM », *Neural computation*, vol. 12, *10*, pp. 2451–2471, 2000, Publisher: MIT press.

[156] M. Sundermeyer, R. Schlüter, and H. Ney, « Lstm neural networks for language modeling. », *in Interspeech*, vol. 2012, 2012, pp. 194–197.

[157] I. Goodfellow *et al.*, « Generative adversarial networks », *Communications of the ACM*, vol. 63, *11*, pp. 139–144, 2020, Publisher: ACM New York, NY, USA.

[158] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, « Generative adversarial networks: an overview », *IEEE signal processing magazine*, vol. 35, *1*, pp. 53–65, 2018, Publisher: IEEE.

[159] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, « BERT: pre-training of deep bidirectional transformers for language understanding », en, *in Proceedings of the 2019 Conference of the North*, Minneapolis, Minnesota: Association for Computational Linguistics, 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. [Online]. Available: http://aclweb.org/anthology/N19-1423 (visited on 06/06/2024).

[160] T. Brown *et al.*, « Language models are few-shot learners », *in Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 1877–1901.

[161] T. Lin, Y. Wang, X. Liu, and X. Qiu, « A survey of transformers », *AI open*, vol. 3, pp. 111–132, 2022, Publisher: Elsevier.

[162] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, and L. Bottou, « Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. », *Journal of machine learning research*, vol. 11, *12*, 2010.

[163] G. E. Hinton, T. J. Sejnowski, *et al.*, « Learning and relearning in boltzmann machines », *Parallel distributed processing: Explorations in the microstructure of cognition*, vol. 1, *282*, p. 2, 1986.

[164] W. S. McCulloch and W. Pitts, « A logical calculus of the ideas immanent in nervous activity », *The bulletin of mathematical biophysics*, vol. 5, pp. 115–133, 1943, Publisher: Springer.

[165] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, « Gradient-based learning applied to document recognition », *Proceedings of the IEEE*, vol. 86, *11*, pp. 2278–2324, 1998, Publisher: Ieee.

[166] K. He, X. Zhang, S. Ren, and J. Sun, « Deep residual learning for image recognition », *in Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[167] W. Hamilton, Z. Ying, and J. Leskovec, « Inductive Representation Learning on Large Graphs », *in Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017. [Online]. Available: `https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html` (visited on 03/15/2023).

[168] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, « Language models are unsupervised multitask learners », *OpenAI blog*, vol. 1, *8*, p. 9, 2019.

[169] J. Achiam *et al.*, « Gpt-4 technical report », *arXiv preprint arXiv:2303.08774*, 2023.

[170] K. Yu, P.-Y. Lung, T. Zhao, P. Zhao, Y.-Y. Tseng, and J. Zhang, « Automatic extraction of protein-protein interactions using grammatical relationship graph », *BMC medical informatics and decision making*, vol. 18, pp. 35–43, 2018, Publisher: Springer.

[171] C. Shi, X. Wang, and P. S. Yu, « Introduction », en, *in Heterogeneous Graph Representation Learning and Applications*, ser. Artificial Intelligence: Foundations, Theory, and Algorithms, C. Shi, X. Wang, and P. S. Yu, Eds., Singapore: Springer, 2022, pp. 1–8, ISBN: 9789811661662. DOI: `10.1007/978-981-16-6166-2_1`. [Online]. Available: `https://doi.org/10.1007/978-981-16-6166-2_1` (visited on 06/22/2023).

[172] L. Page, S. Brin, R. Motwani, and T. Winograd, « The PageRank citation ranking : bringing order to the web », presented at the The Web Conference, Nov. 11, 1999.

[173] S. Bhagat, G. Cormode, and S. Muthukrishnan, « Node classification in social networks », *in Social Network Data Analytics*, C. C. Aggarwal, Ed., Boston, MA: Springer US, 2011, pp. 115–148, ISBN: 978-1-4419-8462-3. DOI: `10.1007/978-1-4419-8462-3_5`. [Online]. Available: `https://doi.org/10.1007/978-1-4419-8462-3_5` (visited on 09/19/2024).

[174] N. Jin, C. Young, and W. Wang, « GAIA: graph classification using evolutionary computation », *in Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, Indianapolis Indiana USA: ACM, Jun. 6, 2010, pp. 879–890, ISBN: 978-1-4503-0032-2. DOI: 10.1145/1807167.1807262. [Online]. Available: https://dl.acm.org/doi/10.1145/1807167.1807262 (visited on 09/19/2024).

[175] O. Younis, M. Krunz, and S. Ramasubramanian, « Node clustering in wireless sensor networks: recent developments and deployment challenges », *IEEE Network*, vol. 20, *3*, pp. 20–25, May 2006, Conference Name: IEEE Network, ISSN: 1558-156X. DOI: 10.1109/MNET.2006.1637928. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1637928 (visited on 09/19/2024).

[176] G. Berlusconi, F. Calderoni, N. Parolini, M. Verani, and C. Piccardi, « Link prediction in criminal networks: a tool for criminal intelligence analysis », *PLOS ONE*, vol. 11, *4*, e0154244, Apr. 22, 2016, Publisher: Public Library of Science, ISSN: 1932-6203. DOI: 10.1371/journal.pone.0154244. [Online]. Available: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0154244 (visited on 09/19/2024).

[177] W. Chen, Y. Wang, and S. Yang, « Efficient influence maximization in social networks », *in Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '09, New York, NY, USA: Association for Computing Machinery, Jun. 28, 2009, pp. 199–208, ISBN: 978-1-60558-495-9. DOI: 10.1145/1557019.1557047. [Online]. Available: https://dl.acm.org/doi/10.1145/1557019.1557047 (visited on 09/19/2024).

[178] B. Perozzi, R. Al-Rfou, and S. Skiena, « Deepwalk: online learning of social representations », *in Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710.

[179] A. Grover and J. Leskovec, « Node2vec: scalable feature learning for networks », *in Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[180] T. N. Kipf and M. Welling, « Semi-Supervised Classification with Graph Convolutional Networks », *in International Conference on Learning Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=SJU4ayYgl.

[181] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, « A comprehensive survey on graph neural networks », *IEEE transactions on neural networks and learning systems*, vol. 32, *1*, pp. 4–24, 2020, Publisher: IEEE.

[182] M. Allamanis, M. Brockschmidt, and M. Khademi, « Learning to represent programs with graphs », *arXiv preprint arXiv:1711.00740*, 2017.

[183] A. LeClair, S. Haque, L. Wu, and C. McMillan, « Improved code summarization via a graph neural network », *in Proceedings of the 28th international conference on program comprehension*, 2020, pp. 184–195.

[184] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, *Graph attention networks*, Feb. 4, 2018. arXiv: 1710.10903[stat]. [Online]. Available: http://arxiv.org/abs/1710.10903 (visited on 10/22/2024).

[185] R. Bommasani *et al.*, *On the opportunities and risks of foundation models*, Jul. 12, 2022. DOI: 10.48550/arXiv.2108.07258. arXiv: 2108.07258. [Online]. Available: http://arxiv.org/abs/2108.07258 (visited on 10/22/2024).

[186] C. Raffel *et al.*, « Exploring the limits of transfer learning with a unified text-to-text transformer », *Journal of Machine Learning Research*, vol. 21, *140*, pp. 1–67, 2020, ISSN: 1533-7928. [Online]. Available: http://jmlr.org/papers/v21/20-074.html (visited on 06/13/2024).

[187] L. Ouyang *et al.*, « Training language models to follow instructions with human feedback », *Advances in neural information processing systems*, vol. 35, pp. 27 730–27 744, 2022.

[188] M. Chen *et al.*, *Evaluating large language models trained on code*, arXiv:2107.03374 [cs], Jul. 14, 2021. DOI: 10.48550/arXiv.2107.03374. arXiv: 2107.03374[cs]. [Online]. Available: http://arxiv.org/abs/2107.03374 (visited on 09/29/2024).

[189] J. Wei *et al.*, « Emergent abilities of large language models », *preprint arXiv:2206.07682*, 2022.

[190] B. Chen, Z. Zhang, N. Langrené, and S. Zhu, *Unleashing the potential of prompt engineering: a comprehensive review*, arXiv:2310.14735 [cs], May 28, 2024. DOI: 10.48550/arXiv.2310.14735. arXiv: 2310.14735[cs]. [Online]. Available: http://arxiv.org/abs/2310.14735 (visited on 06/06/2024).

[191] Z. Zhao, E. Wallace, S. Feng, D. Klein, and S. Singh, « Calibrate Before Use: Improving Few-shot Performance of Language Models », en, *in Proceedings of the 38th International Conference on Machine Learning*, ISSN: 2640-3498, PMLR, Jul. 2021, pp. 12 697–12 706. [Online]. Available: https://proceedings.mlr.press/v139/zhao21c.html (visited on 06/03/2024).

[192] J. Wei *et al.*, « Chain-of-Thought Prompting Elicits Reasoning in Large Language Models », en, *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 824–24 837, Dec. 2022. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html (visited on 06/03/2024).

[193] S. Schulhoff *et al.*, « The prompt report: a systematic survey of prompting techniques », *arXiv preprint arXiv:2406.06608*, 2024.

[194] O. Topsakal and T. C. Akinci, « Creating Large Language Model Applications Utilizing LangChain: A Primer on Developing LLM Apps Fast », en, *International Conference on Applied Engineering and Natural Sciences*, vol. 1, *1*, pp. 1050–1056, Jul. 2023, ISSN: 2980-3209. DOI: 10.59287/icaens.1127. [Online]. Available: https://as-proceeding.com/index.php/icaens/article/view/1127 (visited on 06/07/2024).

[195] D. Riehle, « Composite design patterns », *in Proceedings of the 12th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, ser. OOPSLA '97, New York, NY, USA: Association for Computing Machinery, Oct. 1997, pp. 218–228, ISBN: 978-0-89791-908-1. DOI: 10.1145/263698.263739. [Online]. Available: https://dl.acm.org/doi/10.1145/263698.263739 (visited on 06/07/2024).

[196] S. Kambhampati *et al.*, « Position: LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks », en, Jun. 2024. [Online]. Available: https://openreview.net/forum?id=Th8JPEmH4z (visited on 06/13/2024).

[197] O. Ostapenko *et al.*, *Towards Modular LLMs by Building and Reusing a Library of LoRAs*, en, arXiv:2405.11157 [cs], May 2024. [Online]. Available: http://arxiv.org/abs/2405.11157 (visited on 06/13/2024).

[198] T. A. association, *Automationml*, https://www.automationml.org/, 2022.

[199] T. Leclerc *et al.*, « A flexible and robust framework for the secure systems engineering of space missions », *in 17th International Conference on Space Operations 2023*, Dubai, United Arab Emirates, Mar. 2023. [Online]. Available: https://hal.science/hal-04045293 (visited on 10/09/2024).

[200] D. S. Kolovos, L. M. Rose, N. Drivalos Matragkas, R. F. Paige, F. A. C. Polack, and K. J. Fernandes, « Constructing and navigating non-invasive model decorations », *in Theory and Practice of Model Transformations*, L. Tratt and M. Gogolla, Eds., Berlin, Heidelberg: Springer, 2010, pp. 138–152, ISBN: 978-3-642-13688-7. DOI: 10.1007/978-3-642-13688-7_10.

[201] E. Burger, J. Henss, M. Küster, S. Kruse, and L. Happe, « View-based model-driven software development with ModelJoin », en, *Software & Systems Modeling*, vol. 15, *2*, pp. 473–496, May 1, 2016, ISSN: 1619-1374. DOI: 10.1007/s10270-014-0413-5. [Online]. Available: https://doi.org/10.1007/s10270-014-0413-5 (visited on 06/17/2024).

[202] C. Debreceni, Á. Horváth, Á. Hegedüs, Z. Ujhelyi, I. Ráth, and D. Varró, « Query-driven incremental synchronization of view models », *in Proceedings of the 2nd Workshop on View-Based, Aspect-Oriented and Orthographic Software Modelling*, York United Kingdom: ACM, Jul. 22, 2014, pp. 31–38, ISBN: 978-1-4503-2900-2. DOI: 10.1145/2631675.2631677. [Online]. Available: https://dl.acm.org/doi/10.1145/2631675.2631677 (visited on 10/09/2024).

[203] O. Semeráth, C. Debreceni, Á. Horváth, and D. Varró, « Incremental backward change propagation of view models by logic solvers* », *in Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*, Saint-malo France: ACM, Oct. 2, 2016, pp. 306–316, ISBN: 978-1-4503-4321-3. DOI: 10.1145/2976767.2976788. [Online]. Available: https://dl.acm.org/doi/10.1145/2976767.2976788 (visited on 10/15/2024).

[204] D. Varró, G. Bergmann, Á. Hegedüs, Á. Horváth, I. Ráth, and Z. Ujhelyi, « Road to a reactive and incremental model transformation platform: three generations of the VIATRA framework », *Software & Systems Modeling*, vol. 15, *3*, pp. 609–629, Jul. 1, 2016, ISSN: 1619-1374. DOI: 10.1007/s10270-016-0530-4. [Online]. Available: https://doi.org/10.1007/s10270-016-0530-4 (visited on 10/09/2024).

[205] F. Bedini, R. Maschotta, and A. Zimmermann, « A generative approach for creating eclipse sirius editors for generic systems », *in 2021 IEEE International Systems Conference (SysCon)*, ISSN: 2472-9647, Apr. 2021, pp. 1–8. DOI: 10.1109/SysCon48628.2021.9447062. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9447062 (visited on 10/15/2024).

[206] A. Boronat, « Code-first model-driven engineering: on the agile adoption of MDE tooling », *in 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, ISSN: 2643-1572, Nov. 2019, pp. 874–886. DOI: 10.1109/ASE.2019.00086.

[207] A. Boronat, « EMF-syncer: scalable maintenance of view models over heterogeneous data-centric software systems at run time », *Software and Systems Modeling*, vol. 22, *6*, pp. 1949–1968, Dec. 1, 2023, ISSN: 1619-1374. DOI: 10.1007/s10270-023-01111-7. [Online]. Available: https://doi.org/10.1007/s10270-023-01111-7 (visited on 10/16/2024).

[208] M. El Hamlaoui *et al.*, « A model-driven approach to align heterogeneous models of a complex system. », *The Journal of Object Technology*, vol. 20, *2*, pp. 1–24, 2021, Publisher: Chair of Software Engineering. DOI: 10.5381/jot.2021.20.2.a2. [Online]. Available: https://hal.science/hal-03781930 (visited on 10/07/2024).

[209] H. Klare, M. E. Kramer, M. Langhammer, D. Werle, E. Burger, and R. Reussner, « Enabling consistency in view-based system development — the vitruvius approach », *Journal of Systems and Software*, vol. 171, p. 110 815, Jan. 1, 2021, ISSN: 0164-1212. DOI: 10.1016/j.jss.2020.110815. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121220302144 (visited on 02/03/2022).

[210] N. Weidmann and A. Anjorin, « eMoflon:: neo-consistency and model management with graph databases. », *in STAF workshops*, 2021, pp. 54–64.

[211] C. Atkinson, « Orthographic software modelling: a novel approach to view-based software engineering », *in Modelling Foundations and Applications*, T. Kühne, B. Selic, M.-P. Gervais, and F. Terrier, Eds., Berlin, Heidelberg: Springer, 2010, pp. 1–1, ISBN: 978-3-642-13595-8. DOI: 10.1007/978-3-642-13595-8_1.

[212] M. E. Kramer, E. Burger, and M. Langhammer, « View-centric engineering with synchronized heterogeneous models », *in Proceedings of the 1st Workshop on View-Based, Aspect-Oriented and Orthographic Software Modelling*, ser. VAO '13, New York, NY, USA: Association for Computing Machinery, Jul. 1, 2013, pp. 1–6, ISBN: 978-1-4503-2070-2. DOI: 10.1145/2489861.2489864. [Online]. Available: https://dl.acm.org/doi/10.1145/2489861.2489864 (visited on 10/16/2024).

[213] P. H. Feiler, D. P. Gluch, and J. Hudak, « The architecture analysis & design language (AADL): an introduction », 2006, Publisher: Carnegie Mellon University, Software Engineering Institute.

[214] L. Addazi and F. Ciccozzi, « Blended graphical and textual modelling for UML profiles: a proof-of-concept implementation and experiment », *Journal of Systems and Software*, vol. 175, p. 110 912, May 1, 2021, ISSN: 0164-1212. DOI: 10.1016/j.jss.2021.110912. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121221000091 (visited on 10/16/2024).

[215] I. David *et al.*, « Blended modeling in commercial and open-source model-driven software engineering tools: a systematic study », *Software and Systems Modeling*, vol. 22, *1*, pp. 415–447, Feb. 1, 2023, ISSN: 1619-1374. DOI: 10.1007/s10270-022-01010-3. [Online]. Available: https://doi.org/10.1007/s10270-022-01010-3 (visited on 10/16/2024).

[216] M. Amrani *et al.*, « A survey of federative approaches for model management in MBSE », *in 1st International Workshop on Model Management (MoM) at MODELS 2024*, ACM, Ed., Linz (AUSTRIA), Austria, Sep. 2024. DOI: 10.1145/3652620.3688221. [Online]. Available: https://hal.science/hal-04721128 (visited on 10/16/2024).

[217] L. Burgueño, D. Di Ruscio, H. Sahraoui, and M. Wimmer, « Automation in Model-Driven Engineering: A look back, and ahead », *ACM Trans. Softw. Eng. Methodol.*, Jan. 2025, ISSN: 1049-331X. DOI: 10.1145/3712008. (visited on 01/30/2025).

[218] A. Burdusel, S. Zschaler, and S. John, « Automatic Generation of Atomic Consistency Preserving Search Operators for Search-Based Model Engineering », *in 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS)*, Sep. 2019, pp. 106–116. DOI: 10.1109/MODELS.2019.00-10. (visited on 01/30/2025).

[219] S. John, J. Kosiol, L. Lambers, and G. Taentzer, « A graph-based framework for model-driven optimization facilitating impact analysis of mutation operator properties », *Software and Systems Modeling*, vol. 22, *4*, pp. 1281–1318, Aug. 2023, ISSN: 1619-1374. DOI: 10.1007/s10270-022-01078-x. (visited on 01/30/2025).

[220] G. Kappel, P. Langer, W. Retschitzegger, W. Schwinger, and M. Wimmer, « Model Transformation By-Example: A Survey of the First Wave », en, *in Conceptual Modelling and Its Theoretical Foundations: Essays Dedicated to Bernhard Thalheim on the Occasion of His 60th Birthday*, ser. Lecture Notes in Computer Science, A. Düsterhöft, M. Klettke, and K.-D. Schewe, Eds., Berlin, Heidelberg: Springer, 2012, pp. 197–215, ISBN: 978-3-642-28279-9. DOI: 10.1007/978-3-642-28279-9_15. [Online]. Available: https://doi.org/10.1007/978-3-642-28279-9_15 (visited on 07/07/2023).

[221] M. Strommer, « Model Transformation By-Example », English, Ph.D. dissertation, Vienna University of Technology, Vienna, May 2008. [Online]. Available: https://publik.tuwien.ac.at/files/PubDat_166165.pdf (visited on 07/07/2023).

[222] M. Kessentini, H. Sahraoui, M. Boukadoum, and O. B. Omar, « Search-based model transformation by example », *Software & Systems Modeling*, vol. 11, *2*, pp. 209–226, May 2012, ISSN: 1619-1374. DOI: 10.1007/s10270-010-0175-7. (visited on 01/30/2025).

[223] M. Faunes, H. Sahraoui, and M. Boukadoum, « Genetic-Programming Approach to Learn Model Transformation Rules from Examples », *in Theory and Practice of Model Transformations*, K. Duddy and G. Kappel, Eds., Berlin, Heidelberg: Springer, 2013, pp. 17–32, ISBN: 978-3-642-38883-5. DOI: 10.1007/978-3-642-38883-5_2.

[224] Y. Wang *et al.*, *Agents in software engineering: survey, landscape, and vision*, Sep. 23, 2024. DOI: 10.48550/arXiv.2409.09030. arXiv: 2409.09030. [Online]. Available: http://arxiv.org/abs/2409.09030 (visited on 11/21/2024).

[225] L. Burgueño, R. Clarisó, S. Gérard, S. Li, and J. Cabot, « An nlp-based architecture for the autocompletion of partial domain models », *in CAiSE 2021*, Springer, 2021, pp. 91–106.

[226] J. Di Rocco, C. Di Sipio, P. T. Nguyen, D. Di Ruscio, and A. Pierantonio, « Finding with nemo: a recommender system to forecast the next modeling operations », *in MODELS 2022*, 2022, pp. 154–164.

[227] N. Macedo, T. Jorge, and A. Cunha, « A Feature-Based Classification of Model Repair Approaches », *IEEE Transactions on Software Engineering*, vol. 43, *7*, pp. 615–640, Jul. 2017, Conference Name: IEEE Transactions on Software Engineering, ISSN: 1939-3520. DOI: 10.1109/TSE.2016.2620145.

[228] A. Barriga, R. Heldal, A. Rutle, and L. Iovino, « PARMOREL: a framework for customizable model repair », en, *Software and Systems Modeling*, vol. 21, *5*, pp. 1739–1762, Oct. 1, 2022, ISSN: 1619-1374. DOI: 10.1007/s10270-022-01005-0. [Online]. Available: https://doi.org/10.1007/s10270-022-01005-0 (visited on 11/09/2022).

[229] R. Groner, P. Bellmann, S. Höppner, P. Thiam, F. Schwenker, and M. Tichy, « Predicting the performance of ATL model transformations », *in Proceedings of the 2023 ACM/SPEC International Conference on Performance Engineering*, ser. ICPE '23, Coimbra Portugal: ACM, Apr. 15, 2023, pp. 77–89, ISBN: 9798400700682. DOI: 10.1145/3578244.3583727. [Online]. Available: https://dl.acm.org/doi/10.1145/3578244.3583727 (visited on 10/15/2024).

[230] D.-H. Dang and J. Cabot, « On Automating Inference of OCL Constraints from Counterexamples and Examples », en, *in Knowledge and Systems Engineering*, V.-H. Nguyen, A.-C. Le, and V.-N. Huynh, Eds., ser. Advances in Intelligent Systems and Computing, Cham: Springer International Publishing, 2015, pp. 219–231, ISBN: 978-3-319-11680-8. DOI: 10.1007/978-3-319-11680-8_18.

[231] L. Burgueño, J. Cabot, S. Li, and S. Gérard, « A generic LSTM neural network architecture to infer heterogeneous model transformations », en, *Software and Systems Modeling*, vol. 21, *1*, pp. 139–156, Feb. 2022, ISSN: 1619-1374. DOI: 10.1007/s10270-021-00893-y. [Online]. Available: https://doi.org/10.1007/s10270-021-00893-y (visited on 07/07/2023).

[232] A. Beganovic, M. A. Jaber, and A. A. Almisreb, « Methods and Applications of ChatGPT in Software Development: A Literature Review », en, *Southeast Europe Journal of Soft Computing*, vol. 12, *1*, pp. 08–12, May 2023, Number: 1, ISSN: 2233–1859. [Online]. Available: http://scjournal.ius.edu.ba/index.php/scjournal/article/view/251 (visited on 06/13/2024).

[233] A. Fan *et al.*, *Large Language Models for Software Engineering: Survey and Open Problems*, arXiv:2310.03533 [cs], Nov. 2023. DOI: 10.48550/arXiv.2310.03533. [Online]. Available: http://arxiv.org/abs/2310.03533 (visited on 01/10/2024).

[234] S. G. Bouschery, V. Blazevic, and F. T. Piller, « Augmenting human innovation teams with artificial intelligence: Exploring transformer-based language models », en, *Journal of Product Innovation Management*, vol. 40, *2*, pp. 139–153, 2023, _eprint: https://doi.org/10.1111/jpim ISSN: 1540-5885. DOI: 10.1111/jpim.12656. [Online]. Available: https://doi.org/10.1111/jpim.12656 (visited on 06/13/2024).

[235] M. Alibakhsh, « Challenges of Integrating LLMs Like ChatGPT with Enterprise Software and Solving it with Object Messaging and Intelligent Objects as a New Software Design Paradigm », *in 2023 Congress in Computer Science, Computer Engineering, & Applied Computing (CSCE)*, Jul. 2023, pp. 313–317. DOI: 10.1109/CSCE60160.2023.00054. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10487510 (visited on 06/13/2024).

[236] G. Katsogiannis-Meimarakis and G. Koutrika, « A survey on deep learning approaches for text-to-SQL », en, *The VLDB Journal*, vol. 32, *4*, pp. 905–936, Jul. 2023, ISSN: 0949-877X. DOI: 10.1007/s00778-022-00776-8. [Online]. Available: https://doi.org/10.1007/s00778-022-00776-8 (visited on 06/14/2024).

[237] P. Yin, G. Neubig, W.-t. Yih, and S. Riedel, « TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data », *in Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds., Online: Association for Computational Linguistics, Jul. 2020, pp. 8413–8426. DOI: 10.18653/v1/2020.acl-main.745. [Online]. Available: https://aclanthology.org/2020.acl-main.745 (visited on 06/14/2024).

[238] R. Patil, M. Patwardhan, S. Karande, L. Vig, and G. Shroff, « Exploring Dimensions of Generalizability and Few-shot Transfer for Text-to-SQL Semantic Parsing », en, *in Proceedings of The 1st Transfer Learning for Natural Language Processing Workshop*, ISSN: 2640-3498, PMLR, Jan. 2023, pp. 103–114. [Online]. Available: https://proceedings.mlr.press/v203/patil23a.html (visited on 06/14/2024).

[239] S. Abukhalaf, M. Hamdaqa, and F. Khomh, « On Codex Prompt Engineering for OCL Generation: An Empirical Study », *in 2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*, ISSN: 2574-3864, May 2023, pp. 148–157. DOI: 10.1109/MSR59073.2023.00033. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10173990 (visited on 06/14/2024).

[240] S. Abukhalaf, M. Hamdaqa, and F. Khomh, *PathOCL: Path-Based Prompt Augmentation for OCL Generation with GPT-4*, en, arXiv:2405.12450 [cs], Jun. 2024. [Online]. Available: http://arxiv.org/abs/2405.12450 (visited on 06/14/2024).

[241] J. A. H. López, M. Földiák, and D. Varró, « Text2vql: teaching a model query language to open-source language models with ChatGPT », *in Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems*, ser. MODELS '24, New York, NY, USA: Association for Computing Machinery, Sep. 22, 2024, pp. 13–24, ISBN: 9798400705045. DOI: 10.1145/3640310.3674091. [Online]. Available: https://dl.acm.org/doi/10.1145/3640310.3674091 (visited on 11/21/2024).

[242] R. Gozalo-Brizuela and E. C. Garrido-Merchan, *ChatGPT is not all you need. A State of the Art Review of large Generative AI models*, en, arXiv:2301.04655 [cs], Jan. 2023. [Online]. Available: http://arxiv.org/abs/2301.04655 (visited on 06/16/2024).

[243] T. Händler, « A Taxonomy for Autonomous LLM-Powered Multi-Agent Architectures: » *in Proceedings of the 15th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, Rome, Italy: SCITEPRESS - Science and Technology Publications, 2023, pp. 85–98, ISBN: 978-989-758-671-2. DOI: 10.5220/0012239100003598. [Online]. Available: https://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0012239100003598 (visited on 06/16/2024).

[244] I. Arawjo, C. Swoopes, P. Vaithilingam, M. Wattenberg, and E. L. Glassman, « ChainForge: A Visual Toolkit for Prompt Engineering and LLM Hypothesis Testing », *in Proceedings of the CHI Conference on Human Factors in Computing Systems*, ser. CHI '24, New York, NY, USA: Association for Computing Machinery, May 2024, pp. 1–18, ISBN: 9798400703300. DOI: 10.1145/3613904.3642016. [Online]. Available: https://doi.org/10.1145/3613904.3642016 (visited on 06/16/2024).

[245] T. Schick *et al.*, « Toolformer: Language Models Can Teach Themselves to Use Tools », en, *Advances in Neural Information Processing Systems*, vol. 36, pp. 68 539–68 551, Dec. 2023. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html (visited on 06/16/2024).

[246] A. Singh, A. Ehtesham, S. Mahmud, and J.-H. Kim, « Revolutionizing Mental Health Care through LangChain: A Journey with a Large Language Model », *in 2024 IEEE 14th Annual Computing and Communication Workshop and Conference (CCWC)*, Jan. 2024, pp. 0073–0078. DOI: 10.1109/CCWC60891.2024.10427865. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10427865 (visited on 06/17/2024).

[247] D. Madhav, S. Nijai, U. Patel, and K. Champanerkar, « Question Generation from PDF using LangChain », *in 2024 11th International Conference on Computing for Sustainable Global Development (INDIACom)*, Feb. 2024, pp. 218–222. DOI: 10.23919/INDIACom61295.2024.10499105. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10499105 (visited on 06/17/2024).

[248] O. Cherednichenko, D. Sytnikov, N. Romankiv, N. Sharonova, and P. Sytnikova, « Selection of Large Language Model for development of Interactive Chat Bot for SaaS Solutions », *in 8th International Conference on Computational Linguistics and Intelligent Systems (CoLInS 2024)*, Kharkiv, Ukraine, Apr. 2024. [Online]. Available: https://hal.science/hal-04545073 (visited on 06/17/2024).

[249] M. Dehghani, S. Kolahdouz-Rahimi, M. Tisi, and D. Tamzalit, « Facilitating the migration to the microservice architecture via model-driven reverse engineering and reinforcement learning », en, *Software and Systems Modeling*, vol. 21, *3*, pp. 1115–1133, Jun. 1, 2022, ISSN: 1619-1374. DOI: 10.1007/s10270-022-00977-3. [Online]. Available: https://doi.org/10.1007/s10270-022-00977-3 (visited on 11/09/2022).

[250] X. Tang, Z. Wang, J. Qi, and Z. Li, « Improving code generation from descriptive text by combining deep learning and syntax rules », *Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE*, vol. 2019-July, pp. 385–390, 2019, ISBN: 1891706489, ISSN: 23259086. DOI: 10.18293/SEKE2019-170.

[251] W. Xu, M. Liu, O. Sokolsky, I. Lee, and F. Kong, « LLM-enabled cyber-physical systems: survey, research opportunities, and challenges », en, May 2024, Publisher: International Workshop on Foundation Models for Cyber-Physical Systems & Internet of Things (FMSys). [Online]. Available: https://par.nsf.gov/biblio/10499418-llm-enabled-cyber-physical-systems-survey-research-opportunities-challenges (visited on 06/06/2024).

[252] M. Nejjar, L. Zacharias, F. Stiehle, and I. Weber, *LLMs for science: usage for code generation and data analysis*, arXiv:2311.16733 [cs], Apr. 23, 2024. DOI: 10.48550/arXiv.2311.16733. arXiv: 2311.16733[cs]. [Online]. Available: http://arxiv.org/abs/2311.16733 (visited on 06/03/2024).

[253] G. Bao, H. Zhang, L. Yang, C. Wang, and Y. Zhang, *LLMs with chain-of-thought are non-causal reasoners*, arXiv:2402.16048 [cs], Feb. 25, 2024. DOI: 10.48550/arXiv.2402.16048. arXiv: 2402.16048[cs]. [Online]. Available: http://arxiv.org/abs/2402.16048 (visited on 06/13/2024).

[254] L. Gao *et al.*, « PAL: program-aided language models », en, *in Proceedings of the 40th International Conference on Machine Learning*, ISSN: 2640-3498, PMLR, Jul. 3, 2023, pp. 10 764–10 799. [Online]. Available: https://proceedings.mlr.press/v202/gao23f.html (visited on 06/13/2024).

[255] A. Kansal, « LangChain: your swiss army knife », *in Building generative AI-powered apps: a hands-on guide for developers*, Springer, 2024, pp. 17–40.

[256] Y. Chang *et al.*, « A survey on evaluation of large language models », *ACM Transactions on Intelligent Systems and Technology*, vol. 15, *3*, 39:1–39:45, Mar. 29, 2024, ISSN: 2157-6904. DOI: 10.1145/3641289. [Online]. Available: https://dl.acm.org/doi/10.1145/3641289 (visited on 06/13/2024).

[257] Z. Liu *et al.*, « Information retrieval meets large language models », en, *in Companion Proceedings of the ACM on Web Conference 2024*, Singapore Singapore: ACM, May 13, 2024, pp. 1586–1589, ISBN: 9798400701726. DOI: 10.1145/3589335.3641299. [Online]. Available: https://dl.acm.org/doi/10.1145/3589335.3641299 (visited on 06/13/2024).

[258] S. Ghodratnama and M. Zakershahrak, « Adapting LLMs for efficient, personalized information retrieval: methods and implications », en, *in Service-Oriented Computing – ICSOC 2023 Workshops*, F. Monti *et al.*, Eds., Singapore: Springer Nature, 2024, pp. 17–26, ISBN: 978-981-9709-89-2. DOI: 10.1007/978-981-97-0989-2_2.

[259] J. A. H. López, J. L. Cánovas Izquierdo, and J. S. Cuadrado, « ModelSet: a dataset for machine learning in model-driven engineering », *Software and Systems Modeling*, vol. 21, *3*, pp. 967–986, Jun. 1, 2022, ISSN: 1619-1374. DOI: 10.1007/s10270-021-00929-3. [Online]. Available: https://doi.org/10.1007/s10270-021-00929-3 (visited on 11/09/2022).

[260] S. Yao *et al.*, « ReAct: synergizing reasoning and acting in language models », en, *International Conference on Learning Representations (ICLR)*, Jan. 2023. [Online]. Available: https://par.nsf.gov/biblio/10451467-react-synergizing-reasoning-acting-language-models (visited on 06/11/2024).

[261] L. Zheng *et al.*, *Judging LLM-as-a-judge with MT-bench and chatbot arena*, en, arXiv:2306.05685 [cs], Dec. 23, 2023. arXiv: 2306.05685[cs]. [Online]. Available: http://arxiv.org/abs/2306.05685 (visited on 06/20/2024).

[262] D. Liben-Nowell and J. Kleinberg, « The link prediction problem for social networks », *in Proceedings of the twelfth international conference on Information and knowledge management*, ser. CIKM '03, New York, NY, USA: Association for Computing Machinery, Nov. 2003, pp. 556–559, ISBN: 978-1-58113-723-1. DOI: 10.1145/956863.956972. [Online]. Available: https://dl.acm.org/doi/10.1145/956863.956972 (visited on 06/23/2023).

[263] B. Perozzi, R. Al-Rfou, and S. Skiena, « DeepWalk: online learning of social representations », *in Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '14, New York, NY, USA: Association for Computing Machinery, Aug. 2014, pp. 701–710, ISBN: 978-1-4503-2956-9. DOI: 10.1145/2623330.2623732. [Online]. Available: https://dl.acm.org/doi/10.1145/2623330.2623732 (visited on 06/23/2023).

[264] L. Lü, M. Medo, C. H. Yeung, Y.-C. Zhang, Z.-K. Zhang, and T. Zhou, « Recommender systems », en, *Physics Reports*, Recommender Systems, vol. 519, *1*, pp. 1–49, Oct. 2012, ISSN: 0370-1573. DOI: 10.1016/j.physrep.2012.02.006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0370157312000828 (visited on 06/23/2023).

[265] C. Shi, X. Wang, and P. S. Yu, « Platforms and Practice of Heterogeneous Graph Representation Learning », en, *in Heterogeneous Graph Representation Learning and Applications*, ser. Artificial Intelligence: Foundations, Theory, and Algorithms, C. Shi, X. Wang, and P. S. Yu, Eds., Singapore: Springer, 2022, pp. 285–310, ISBN: 9789811661662. DOI: 10.1007/978-981-16-6166-2_10. [Online]. Available: https://doi.org/10.1007/978-981-16-6166-2_10 (visited on 06/27/2023).

[266] F. M. Harper and J. A. Konstan, « The MovieLens Datasets: History and Context », en, *ACM Transactions on Interactive Intelligent Systems*, vol. 5, *4*, pp. 1–19, Jan. 2016, ISSN: 2160-6455, 2160-6463. DOI: 10.1145/2827872. [Online]. Available: https://dl.acm.org/doi/10.1145/2827872 (visited on 06/15/2023).

[267] K. K. Jena *et al.*, « Neural model based collaborative filtering for movie recommendation system », en, *International Journal of Information Technology*, vol. 14, *4*, pp. 2067–2077, Jun. 2022, ISSN: 2511-2112. DOI: 10.1007/s41870-022-00858-4. [Online]. Available: https://doi.org/10.1007/s41870-022-00858-4 (visited on 09/08/2023).

[268] A. Akbar, P. Agarwal, and A. J. Obaid, « Recommendation engines-neural embedding to graph-based: Techniques and evaluations », *International Journal of Nonlinear Analysis and Applications*, vol. 13, *1*, pp. 2411–2423, Mar. 2022, Publisher: Semnan University, ISSN: 2008-6822. DOI: 10.22075/ijnaa.2022.5941. [Online]. Available: https://ijnaa.semnan.ac.ir/article_5941.html (visited on 09/08/2023).

[269] M. S. Kumar and J. Prabhu, « A hybrid model collaborative movie recommendation system using K-means clustering with ant colony optimisation », *International Journal of Internet Technology and Secured Transactions*, vol. 10, *3*, pp. 337–354, Jan. 2020, Publisher: Inderscience Publishers, ISSN: 1748-569X. DOI: 10.1504/IJITST.2020.107079. [Online]. Available: https://www.inderscienceonline.com/doi/abs/10.1504/IJITST.2020.107079 (visited on 09/08/2023).

[270] N. Reimers and I. Gurevych, « Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks », *in Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3982–3992. DOI: 10.18653/v1/D19-1410. [Online]. Available: https://aclanthology.org/D19-1410 (visited on 06/29/2023).

[271] T. Fawcett, « An introduction to ROC analysis », en, *Pattern Recognition Letters*, ROC Analysis in Pattern Recognition, vol. 27, *8*, pp. 861–874, Jun. 2006, ISSN: 0167-8655. DOI: 10.1016/j.patrec.2005.10.010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016786550500303X (visited on 06/28/2023).

[272] E. Guerra, J. de Lara, D. S. Kolovos, and R. F. Paige, « Inter-modelling: from theory to practice », en, *in Model Driven Engineering Languages and Systems*, D. C. Petriu, N. Rouquette, and y. Haugen, Eds., ser. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2010, pp. 376–391, ISBN: 978-3-642-16145-2. DOI: 10.1007/978-3-642-16145-2_26.

[273] J. A. H. López, J. S. Cuadrado, R. Rubei, and D. Di Ruscio, « ModelXGlue: a benchmarking framework for ML tools in MDE », *Software and Systems Modeling*, Jun. 10, 2024, ISSN: 1619-1374. DOI: 10.1007/s10270-024-01183-z. [Online]. Available: https://doi.org/10.1007/s10270-024-01183-z (visited on 11/21/2024).

[274] L. Dai, B. Wang, W. Xiang, and Y. Mo, *Bi-directional iterative prompt-tuning for event argument extraction*, Oct. 28, 2022. DOI: 10.48550/arXiv.2210.15843. arXiv: 2210.15843. [Online]. Available: http://arxiv.org/abs/2210.15843 (visited on 11/21/2024).

[275] Y. Zhang, Y. Yuan, and A. C.-C. Yao, *On the diagram of thought*, Sep. 16, 2024. DOI: 10.48550/arXiv.2409.10038. arXiv: 2409.10038. [Online]. Available: http://arxiv.org/abs/2409.10038 (visited on 11/21/2024).

[276] Z. Sprague *et al.*, *To CoT or not to CoT? chain-of-thought helps mainly on math and symbolic reasoning*, Oct. 29, 2024. DOI: 10.48550/arXiv.2409.12183. arXiv: 2409.12183. [Online]. Available: http://arxiv.org/abs/2409.12183 (visited on 11/21/2024).

[277] T. Liu *et al.*, *Logic-of-thought: injecting logic into contexts for full reasoning in large language models*, Sep. 26, 2024. DOI: 10.48550/arXiv.2409.17539. arXiv: 2409.17539. [Online]. Available: http://arxiv.org/abs/2409.17539 (visited on 11/21/2024).

[278] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, « Pre-train, prompt, and predict: a systematic survey of prompting methods in natural language processing », *ACM Comput. Surv.*, vol. 55, *9*, 195:1–195:35, Jan. 16, 2023, ISSN: 0360-0300. DOI: 10.1145/3560815. [Online]. Available: https://dl.acm.org/doi/10.1145/3560815 (visited on 11/21/2024).

[279] P. Lewis *et al.*, *Retrieval-augmented generation for knowledge-intensive NLP tasks*, Apr. 12, 2021. DOI: 10.48550/arXiv.2005.11401. arXiv: 2005.11401. [Online]. Available: http://arxiv.org/abs/2005.11401 (visited on 11/21/2024).

[280] J. Liu, X. Han, C. Deng, and J. Feng, « Improving self-consistency for open-domain question answering via automatic prompt engineering and ensemble learning », *in Natural Language Processing and Chinese Computing*, D. F. Wong, Z. Wei, and M. Yang, Eds., Singapore: Springer Nature, 2025, pp. 359–371, ISBN: 978-981-9794-34-8. DOI: 10.1007/978-981-97-9434-8_28.

[281] H.-J. Ye, D.-C. Zhan, Y. Jiang, and Z.-H. Zhou, « Heterogeneous few-shot model rectification with semantic mapping », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, *11*, pp. 3878–3891, Nov. 2021, Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, ISSN: 1939-3539. DOI: 10.1109/TPAMI.2020.2994749. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9093972 (visited on 11/21/2024).

[282] Q. Wu *et al.*, *AutoGen: enabling next-gen LLM applications via multi-agent conversation*, Oct. 3, 2023. DOI: 10.48550/arXiv.2308.08155. arXiv: 2308.08155. [Online]. Available: http://arxiv.org/abs/2308.08155 (visited on 11/21/2024).

[283] Z. Schillaci, « LLM Adoption Trends and Associated Risks », en, *in Large Language Models in Cybersecurity: Threats, Exposure and Mitigation*, A. Kucharavy, O. Plancherel, V. Mulder, A. Mermoud, and V. Lenders, Eds., Cham: Springer Nature Switzerland, 2024, pp. 121–128, ISBN: 978-3-031-54827-7. DOI: 10.1007/978-3-031-54827-7_13. [Online]. Available: https://doi.org/10.1007/978-3-031-54827-7_13 (visited on 06/17/2024).

[284] E. Mosqueira-Rey, E. Hernández-Pereira, D. Alonso-Ríos, J. Bobes-Bascarán, and Á. Fernández-Leal, « Human-in-the-loop machine learning: a state of the art », *Artificial Intelligence Review*, vol. 56, *4*, pp. 3005–3054, Apr. 1, 2023, ISSN: 1573-7462. DOI: 10.1007/s10462-022-10246-w. [Online]. Available: https://doi.org/10.1007/s10462-022-10246-w (visited on 11/21/2024).

[285] J. Giner-Miguelez, A. Gómez, and J. Cabot, « A domain-specific language for describing machine learning datasets », en, *Journal of Computer Languages*, vol. 76, p. 101 209, Aug. 2023, ISSN: 2590-1184. DOI: 10.1016/j.cola.2023.101209. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2590118423000199 (visited on 07/07/2023).

[286] Z. Rajaei, S. Kolahdouz-Rahimi, M. Tisi, and F. Jouault, « A DSL for Encoding Models for Graph-Learning Processes », en, Jun. 2021. [Online]. Available: https://hal.science/hal-03252919 (visited on 07/07/2023).

[287] O. O. Koyejo, N. Natarajan, P. K. Ravikumar, and I. S. Dhillon, « Consistent Binary Classification with Generalized Performance Metrics », *in Advances in Neural Information Processing Systems*, vol. 27, Curran Associates, Inc., 2014.

[288] Z. Zhong, C.-T. Li, and J. Pang, « Personalised meta-path generation for heterogeneous graph neural networks », *Data Mining and Knowledge Discovery*, vol. 36, *6*, pp. 2299–2333, Nov. 1, 2022, ISSN: 1573-756X. DOI: 10.1007/s10618-022-00862-z. [Online]. Available: https://doi.org/10.1007/s10618-022-00862-z (visited on 11/17/2024).

[289] A. Marcén, A. Iglesias, R. Lapeña Martí, F. Pérez, and C. Cetina, « A systematic literature review of model-driven engineering using machine learning », *IEEE Transactions on Software Engineering*, vol. PP, pp. 1–25, Sep. 1, 2024. DOI: 10.1109/TSE.2024.3430514.

[290] S. Uchitel *et al.*, « Scoping software engineering for AI: the TSE perspective », *IEEE Transactions on Software Engineering*, vol. 50, *11*, pp. 2709–2711, Nov. 2024, Conference Name: IEEE Transactions on Software Engineering, ISSN: 1939-3520. DOI: 10.1109/TSE.2024.3470368. [Online]. Available: https://ieeexplore.ieee.org/document/10752650/?arnumber=10752650 (visited on 11/21/2024).

[291] S. Rädler, L. Berardinelli, K. Winter, A. Rahimi, and S. Rinderle-Ma, « Bridging MDE and AI: a systematic review of domain-specific languages and model-driven practices in AI software systems engineering », *Software and Systems Modeling*, Sep. 28, 2024, ISSN: 1619-1374. DOI: 10.1007/s10270-024-01211-y. [Online]. Available: https://doi.org/10.1007/s10270-024-01211-y (visited on 11/21/2024).

[292] Y. Tian *et al.*, « Graph neural prompting with large language models », *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, *17*, pp. 19 080–19 088, Mar. 24, 2024, Number: 17, ISSN: 2374-3468. DOI: 10.1609/aaai.v38i17.29875. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/29875 (visited on 11/02/2024).

[293] R. Orlando, P.-L. Huguet Cabot, E. Barba, and R. Navigli, « ReLiK: retrieve and LinK, fast and accurate entity linking and relation extraction on an academic budget », *in Findings of the Association for Computational Linguistics: ACL 2024*, L.-W. Ku, A. Martins, and V. Srikumar, Eds., Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 14 114–14 132. DOI: 10.18653/v1/2024.findings-acl.839. [Online]. Available: https://aclanthology.org/2024.findings-acl.839 (visited on 11/02/2024).

[294] X. Ma, G. Fang, and X. Wang, « LLM-pruner: on the structural pruning of large language models », *Advances in Neural Information Processing Systems*, vol. 36, pp. 21 702–21 720, Dec. 15, 2023. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/hash/44956951349095f74492a5471128a7e0-Abstract-Conference.html (visited on 11/21/2024).

[295] Y. Guo, Y. Lang, and Q. Ren, « GPTQT: quantize large language models twice to push the efficiency », *in 2024 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE International Conference on Robotics, Automation and Mechatronics (RAM)*, ISSN: 2326-8239, Aug. 2024, pp. 368–373. DOI: 10.1109/CIS-RAM61939.2024.10672819. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10672819 (visited on 11/21/2024).

[296] T. McDonald and A. Emami, « Trace-of-thought prompting: investigating prompt-based knowledge distillation through question decomposition », *in Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, X. Fu and E. Fleisig, Eds., Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 293–306. DOI: 10.18653/v1/2024.acl-srw.35. [Online]. Available: https://aclanthology.org/2024.acl-srw.35 (visited on 11/21/2024).

[297] H. Ma, Y. Rong, and J. Huang, « Graph neural networks: scalability », *in Graph Neural Networks: Foundations, Frontiers, and Applications*, L. Wu, P. Cui, J. Pei, and L. Zhao, Eds., Singapore: Springer Nature, 2022, pp. 99–119, ISBN: 9789811660542. DOI: 10.1007/978-981-16-6054-2_6. [Online]. Available: https://doi.org/10.1007/978-981-16-6054-2_6 (visited on 11/21/2024).

[298] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, « On the dangers of stochastic parrots: can language models be too big? », en, *in Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, ser. FAccT '21, New York, NY, USA: Association for Computing Machinery, Mar. 1, 2021, pp. 610–623, ISBN: 978-1-4503-8309-7. DOI: 10.1145/3442188.3445922. [Online]. Available: https://dl.acm.org/doi/10.1145/3442188.3445922 (visited on 11/02/2024).

[299] M. Buyl *et al.*, *Large language models reflect the ideology of their creators*, Oct. 24, 2024. DOI: 10.48550/arXiv.2410.18417. arXiv: 2410.18417. [Online]. Available: http://arxiv.org/abs/2410.18417 (visited on 11/02/2024).

COLLEGES SCIENCES
BRETAGNE POUR L'INGENIEUR
LOIRE ET LE NUMERIQUE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Fédération de Modèles Hétérogènes avec des Vues sur les Modèles Assistées par l'Apprentissage Automatique

Mot clés : Ingénierie dirigée par les modèles, Vues sur les modèles, Grands modèles de langage, Ingénierie des prompts, Réseaux de neurones en graphes, Apprentissage profond

Résumé : L'Ingénierie Dirigée par les Modèles (IDM) promeut les modèles comme un élément clé pour répondre à la complexité croissante du cycle de vie des systèmes logiciel. L'ingénierie de systèmes avec l'IDM implique divers modèles représentant différentes aspects du système. Cette hétérogénéité nécessite des capacités de fédération de modèles pour intégrer des points de vue spécifiques à de multiples domaines. Les solutions de Vues sur les Modèles (Model Views) répondent à ce défi mais manquent encore de support à l'automatisation. Cette thèse explore l'intégration de l'*Apprentissage Automatique (AA)*, notamment les *Réseaux de Neurones en Graphes (GNN)* et *Grands Modèles de Langage (LLM)*, pour améliorer la définition et construction de telles vues. La solution proposée introduit une approche en deux volets dans la solution technique *EMF Views*. Cela a permis d'automatiser partiellement la définition des vues sur modèles à la conception, et de calculer dynamiquement les liens inter-modèles à l'exécution. Nos résultats indiquent que l'application de techniques d'apprentissage profond (DL), dans ce contexte spécifique de l'IDM, permet déjà d'atteindre un premier niveau d'automatisation intéressant. Plus globalement, cet effort de recherche contribue au développement actuel de solutions plus intelligentes pour l'IDM.

Title: Federation of Heterogeneous Models with Machine Learning-Assisted Model Views

Keywords: Model-driven engineering, Model views, Large language models, Prompt engineering, Graph Neural Networks, Deep Learning

Abstract: Model-driven engineering (MDE) promotes models as a key element in addressing the increasing complexity of the software systems' lifecycle. Engineering systems with MDE involves various models representing different system aspects. This heterogeneity requires model federation capabilities to integrate viewpoints specific to multiple domains. Model View solutions address this challenge but still lack more automation support. This thesis explores the integration of *Machine Learning (ML)*, notably *Graph Neural Networks (GNNs)* and *Large Language Models (LLMs)*, in order to improve the definition and building of such views. The proposed solution introduces a twofold approach within the *EMF Views* technical solution. This allowed to partially automate the definition of model views at design time, and to dynamically compute inter-model links at runtime. Our results indicate that the application of Deep Learning (DL) techniques, in this particular MDE context, already allows to achieve a first relevant level of automation. More globally, this research effort contributes to the ongoing development of more intelligent MDE solutions.