COLLEGE MATHS, TELECOMS
DOCTORAL INFORMATIQUE, SIGNAL
BRETAGNE SYSTEMES, ELECTRONIQUE

Université de Rennes

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES

ÉCOLE DOCTORALE N° 601
Mathématiques, Télécommunications, Informatique, Signal, Systèmes, Électronique
Spécialité : *« voir README et le site de votre école doctorale »*

Par

« Prénom NOM »

« Titre de la thèse »

« Sous-titre de la thèse »

Thèse présentée et soutenue à « Lieu », le « date »
Unité de recherche : « voir README et le site de de votre école doctorale »
Thèse N° : « si pertinent »

Rapporteurs avant soutenance :

Prénom NOM Fonction et établissement d'exercice
Prénom NOM Fonction et établissement d'exercice
Prénom NOM Fonction et établissement d'exercice

Composition du Jury :

Attention, en cas d'absence d'un des membres du Jury le jour de la soutenance, la composition du jury doit être revue pour s'assurer qu'elle est conforme et devra être répercutée sur la couverture de thèse

Président :	Prénom NOM	Fonction et établissement d'exercice *(à préciser après la soutenance)*
Examinateurs :	Prénom NOM	Fonction et établissement d'exercice
	Prénom NOM	Fonction et établissement d'exercice
	Prénom NOM	Fonction et établissement d'exercice
	Prénom NOM	Fonction et établissement d'exercice
Dir. de thèse :	Prénom NOM	Fonction et établissement d'exercice
Co-dir. de thèse :	Prénom NOM	Fonction et établissement d'exercice *(si pertinent)*

Invité(s) :

Prénom NOM Fonction et établissement d'exercice

Acknowledgement

Je tiens à remercier
I would like to thank. my parents..
J'adresse également toute ma reconnaissance à ....
....

Table of Contents

Introduction **7**

Context and Motivation 7

Challenges 8

Contributions 9

Bibliography **11**

INTRODUCTION

Context and Motivation

Software systems are exponentially growing, which leads to a substantial burden in terms of maintenance, often resulting in a high cost that may surpass the cost of software development itself [4]. Since Object Management Group (OMG) has introduced Model driven Engineering in 2001 [2], MDE has been prominent in developing and maintaining large-scale and embedded systems while increasing the developers' productivity. By adopting MDE, industry can reduce time (development time, time-to-market), costs (development, integration, reconfiguration), and improve sustainability and international competitiveness [11, 26]. MDE raises the abstraction level to the "metamodel" artifact in order to separate the implementation from the business logic. Metamodel is a central artifact for building software languages [3]. It specifies the domain concepts, their properties, and the relationship between them. A metamodel is the cornerstone to generate model instances, constraints, transformations, and code when building the necessary language tooling, e.g. editor, checker, compiler, data access layers, etc. In particular, metamodels are used as inputs for complex code generators that leverage on the abstract concepts defined in metamodels. The generated code API for creating, loading and manipulating the model instances, adapters, serialization facilities, and an editor, all from the metamodel elements. This generated code is further enriched by developers to offer additional functionalities and tooling, such as validation, transformation, simulation, or debugging. For instance, UML [1] and BPMN [2] Eclipse implementations rely on the UML and BPMN metamodels to generate their corresponding code API before building around it all their tooling and services in the additional code.

1. `https://www.eclipse.org/modeling/mdt/downloads/?project=uml2`
2. `https://www.eclipse.org/bpmn2-modeler/`

Challenges

C1: Resolve the impact of the metamodel evolution on the code automatically

One of the foremost challenges to deal with in MDE is the impact of the evolution of metamodels on its dependent artifacts. We we focus on the impact of metamodels' evolution on the code. Indeed, when a metamodel evolves and the core API is regenerated again, the additional code implemented by developers can be impacted. As a consequence, this additional code must be co-evolved accordingly by executing a resolution for each impacted part of the code.

However, manual co-evolution can be tedious, error-prone, and time-consuming. Therefore, it is essential to support an automatic co-evolution of code when metamodels evolve. The co-evolution challenge has been extensively addressed in *MDE*. In particular, [24, 14, 23, 13, 12, 28] focused on consistency checking between models and code, but not its co-evolution. Other works [27, 20] proposed to co-evolve the code. However, the former handles only the generated code API, it does not handle additional code and aims to maintain bidirectional traceability between the model and the code API. The latter supports a semi-automatic co-evolution requiring developers' intervention. Moreover, it does not use any validation process to check the correctness of the co-evolution and with no comparison to a baseline.

Considering that metamodel and co-evolution is one of many other MDE tasks like for example Model generation, code generation. Since their appearance, LLMs have been applied in different domains of scientific research, such as Software Engineering and Model-Driven Engineering (MDE), however, the challenge of exploring LLMs in the task of metamodel and code co-evolution is never addressed.

C2: Behavioral correctness of the metamodel and code co-evolution

In literature, when the problem of metamodel and code co-evolution is addressed, the challenge of checking that the co-evolution impacted or not the behavioral correctness of the code is not handled. In any Model-driven engineered system, the elements of the metamodel are used un the code. The evolution of the metamodel will be propagated in the code that is co-evolved and its behavior may be altered. Hence, the importance of checking the correctness of the co-evolution.

Contributions

To tackle these challenges, we propose three contributions :

— First, we propose a fully automatic code co-evolution approach du to metamodel evolution based on pattern matching. Our approach handles both atomic and complex changes (ref) of the metamodel.

— Second, we investigate the ability of LLMs in giving correct co-evolutions in the context of metamodel and code co-evolution

— Third, w propose an approach that assist developers to check the behavioral correctness of the co-evolution. This approach leverages unit tests before and after the co-evolution and gives visual report about passing, failing, and errounous tests before and after the co-evolution.

BIBLIOGRAPHY

[1] Edouard Batot et al., « Heuristic-Based Recommendation for Metamodel—OCL Coevolution », *in*: *2017 ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, IEEE, 2017, pp. 210–220.

[2] Marco Brambilla, Jordi Cabot, and Manuel Wimmer, *Model-driven software engineering in practice*, Morgan & Claypool Publishers, 2017.

[3] Jordi Cabot and Martin Gogolla, « Object constraint language (OCL): a definitive guide », *in*: *Formal methods for model-driven engineering*, Springer, 2012, pp. 58–90.

[4] Yi-Ting Chen, Chin-Yu Huang, and Tsung-Han Yang, « Using multi-pattern clustering methods to improve software maintenance quality », *in*: *IET Software* 17.*1* (2023), pp. 1–22, DOI: https://doi.org/10.1049/sfw2.12075, eprint: https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/sfw2.12075, URL: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/sfw2.12075.

[5] Antonio Cicchetti et al., « Automating co-evolution in model-driven engineering », *in*: *2008 12th International IEEE enterprise distributed object computing conference*, IEEE, 2008, pp. 222–231.

[6] Alexandre Correa and Cláudia Werner, « Refactoring object constraint language specifications », *in*: *Software & Systems Modeling* 6.*2* (2007), pp. 113–138.

[7] Kelly Garcés et al., « Adapting transformations to metamodel changes via external transformation composition », *in*: *Software & Systems Modeling* 13.*2* (2014), pp. 789–806.

[8] Kelly Garcés et al., « Managing model adaptation by precise detection of metamodel changes », *in*: *Model Driven Architecture-Foundations and Applications: 5th European Conference, ECMDA-FA 2009, Enschede, The Netherlands, June 23-26, 2009. Proceedings 5*, Springer, 2009, pp. 34–49.

[9] Jokin Garcıa, Oscar Diaz, and Maider Azanza, « Model transformation co-evolution: A semi-automatic approach », *in*: *Software Language Engineering: 5th International Conference, SLE 2012, Dresden, Germany, September 26-28, 2012, Revised Selected Papers 5*, Springer, 2013, pp. 144–163.

[10] Markus Herrmannsdoerfer, Sebastian Benz, and Elmar Juergens, « COPE-Automating Coupled Evolution of Metamodels and Models. », *in*: *ECOOP*, vol. 9, Springer, 2009, pp. 52–76.

[11] John Hutchinson et al., « Empirical assessment of MDE in industry », *in*: *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, Waikiki, Honolulu, HI, USA: Association for Computing Machinery, 2011, pp. 471–480, ISBN: 9781450304450, DOI: 10.1145/1985793.1985858, URL: https://doi.org/10.1145/1985793.1985858.

[12] Robbert Jongeling et al., « Structural consistency between a system model and its implementation: a design science study in industry », *in*: *The European Conference on Modelling Foundations and Applications (ECMFA)*, 2022.

[13] Robbert Jongeling et al., « Towards consistency checking between a system model and its implementation », *in*: *International Conference on Systems Modelling and Management*, Springer, 2020, pp. 30–39.

[14] Georgios Kanakis et al., « An Empirical Study on the Impact of Inconsistency Feedback during Model and Code Co-changing. », *in*: *The Journal of Object Technology* 18.*2* (2019), pp. 10–1.

[15] Wael Kessentini, Houari Sahraoui, and Manuel Wimmer, « Automated Co-evolution of Metamodels and Transformation Rules: A Search-Based Approach », *in*: *International Symposium on Search Based Software Engineering*, Springer, 2018, pp. 229–245.

[16] Wael Kessentini, Houari Sahraoui, and Manuel Wimmer, « Automated metamodel/model co-evolution: A search-based approach », *in*: *Information and Software Technology* 106 (2019), pp. 49–67.

[17] Wael Kessentini, Manuel Wimmer, and Houari Sahraoui, « Integrating the Designer in-the-loop for Metamodel/Model Co-Evolution via Interactive Computational Search », *in*: *Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, ACM, 2018, pp. 101–111.

[18] Djamel Eddine Khelladi, Roland Kretschmer, and Alexander Egyed, « Change Propagation-based and Composition-based Co-evolution of Transformations with Evolving Metamodels », *in*: *Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, ACM, 2018, pp. 404–414.

[19] Djamel Eddine Khelladi et al., « A semi-automatic maintenance and co-evolution of OCL constraints with (meta) model evolution », *in*: *Journal of Systems and Software* 134 (2017), pp. 242–260.

[20] Djamel Eddine Khelladi et al., « Co-Evolving Code with Evolving Metamodels », *in*: *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ICSE '20, Seoul, South Korea: Association for Computing Machinery, 2020, pp. 1496–1508, ISBN: 9781450371216, DOI: 10.1145/3377811.3380324, URL: https://doi.org/10.1145/3377811.3380324.

[21] Angelika Kusel et al., « Consistent co-evolution of models and transformations », *in*: *ACM/IEEE 18th MODELS*, 2015, pp. 116–125.

[22] Angelika Kusel et al., « Systematic Co-Evolution of OCL Expressions », *in*: *11th APCCM 2015*, vol. 27, 2015, p. 30.

[23] Van Cam Pham et al., « Bidirectional mapping between architecture model and code for synchronization », *in*: *2017 IEEE International Conference on Software Architecture (ICSA)*, IEEE, 2017, pp. 239–242.

[24] Markus Riedl-Ehrenleitner, Andreas Demuth, and Alexander Egyed, « Towards model-and-code consistency checking », *in*: *2014 IEEE 38th Annual Computer Software and Applications Conference*, IEEE, 2014, pp. 85–90.

[25] Guido Wachsmuth, « Metamodel adaptation and model co-adaptation », *in*: *ECOOP*, vol. 7, Springer, 2007, pp. 600–624.

[26] Andreas Wortmann et al., « Modeling languages in Industry 4.0: an extended systematic mapping study », *in*: *Softw. Syst. Model.* 19.*1* (Jan. 2020), pp. 67–94, ISSN: 1619-1366, DOI: 10.1007/s10270-019-00757-6, URL: https://doi.org/10.1007/s10270-019-00757-6.

[27] Yijun Yu et al., « Maintaining invariant traceability through bidirectional transformations », *in*: *2012 34th International Conference on Software Engineering (ICSE)*, IEEE, 2012, pp. 540–550.

[28] MohammadAmin Zaheri, Michalis Famelis, and Eugene Syriani, « Towards Checking Consistency-Breaking Updates between Models and Generated Artifacts », *in*: *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, IEEE, 2021, pp. 400–409.

COLLEGE DOCTORAL BRETAGNE
MATHS, TELECOMS
INFORMATIQUE, SIGNAL
SYSTEMES, ELECTRONIQUE

Université de Rennes

Titre : titre (en français).....

Mot clés : de 3 à 6 mots clefs

Résumé : Eius populus ab incunabulis primis ad usque pueritiae tempus extremum, quod annis circumcluditur fere trecentis, circummurana pertulit bella, deinde aetatem ingressus adultam post multiplices bellorum aerumnas Alpes transcendit et fretum, in iuvenem erectus et virum ex omni plaga quam orbis ambit inmensus, reportavit laureas et triumphos, iamque vergens in senium et nomine solo aliquotiens vincens ad tranquilliora vitae discessit. Hoc inmaturo interitu ipse quoque sui pertaesus excessit e vita aetatis nono anno atque vicensimo cum quadriennio imperasset. natus apud Tuscos in Massa Veternensi, patre Constantio Constantini fratre imperatoris, matreque Galla. Thalassius vero ea tempestate praefectus praetorio praesens ipse quoque adrogantis ingenii, considerans incitationem eius ad multorum augeri discrimina, non maturitate vel consiliis mitigabat, ut aliquotiens celsae potestates iras principum molliverunt, sed adversando iurgandoque cum parum congrueret, eum ad rabiem potius evibrabat, Augustum actus eius exaggerando creberrime docens, idque, incertum qua mente, ne lateret adfectans. quibus mox Caesar acrius efferatus, velut contumaciae quoddam vexillum altius erigens, sine respectu salutis alienae vel suae ad vertenda opposita instar rapidi fluminis irrevocabili impetu ferebatur. Hae duae provinciae bello quondam piratico catervis mixtae praedonum.

Title: titre (en anglais).....

Keywords: de 3 à 6 mots clefs

Abstract: Eius populus ab incunabulis primis ad usque pueritiae tempus extremum, quod annis circumcluditur fere trecentis, circummurana pertulit bella, deinde aetatem ingressus adultam post multiplices bellorum aerumnas Alpes transcendit et fretum, in iuvenem erectus et virum ex omni plaga quam orbis ambit inmensus, reportavit laureas et triumphos, iamque vergens in senium et nomine solo aliquotiens vincens ad tranquilliora vitae discessit. Hoc inmaturo interitu ipse quoque sui pertaesus excessit e vita aetatis nono anno atque vicensimo cum quadriennio imperasset. natus apud Tuscos in Massa Veternensi, patre Constantio Constantini fratre imperatoris, matreque Galla. Thalassius vero ea tempestate praefectus praetorio praesens ipse quoque adrogantis ingenii, considerans incitationem eius ad multorum augeri discrimina, non maturitate vel consiliis mitigabat, ut aliquotiens celsae potestates iras principum molliverunt, sed adversando iurgandoque cum parum congrueret, eum ad rabiem potius evibrabat, Augustum actus eius exaggerando creberrime docens, idque, incertum qua mente, ne lateret adfectans. quibus mox Caesar acrius efferatus, velut contumaciae quoddam vexillum altius erigens, sine respectu salutis alienae vel suae ad vertenda opposita instar rapidi fluminis irrevocabili impetu ferebatur. Hae duae provinciae bello quondam piratico catervis mixtae praedonum.