



École nationale Supérieure d’Informatique
École Nationale Supérieure de Mécanique et d’Aérotechnique

THESE

pour l’obtention du Grade de
DOCTEUR DE L'ESI

Ecole Doctorale
Secteur de recherche : informatique

Présentée par :

Abdelkader OUARED

**Vers un langage de description, un système de
persistance et un processus de réutilisation des modèles
de coût des bases de données**

Directeurs de Thèse : Ladjel BELLATRECHE , Kamel BOUKHALFA

Soutenue le 24/06/2018
devant la Commission d’Examen

JURY

Président:

Rapporteurs:

Examinateurs:

Directeurs :	Ladjel BELLATRECHE	Professeur, ISAE-ENSMA, Poitiers
	Kamel BOUKHALFA	Professeur, USTHB, Alger

Résumé

En raison de l'intérêt croissant pour les techniques de développement de systèmes pilotées par des modèles, la conception efficace de transformations automatisées de modèles entre des modèles hétérogènes est devenue un défi majeur dans le développement de logiciels. Alors qu'un certain nombre de langages spécialisés ont été proposés, visant à spécifier des transformations de modèles, il n'existe actuellement aucune base mûrie pour spécifier des transformations entre de tels modèles qui reposent sur une collaboration concepteurs/experts afin de proposer une solution qui satisfasse les exigences des utilisateurs et contraintes applicatives. Ce processus de transformation est une tâche complexe et doit imiter la manière dont les concepteurs et les experts aux perspectives différentes se comportent et réfléchissent aux transformations de modèles. Dans cet article, nous proposons un cadre basé sur une nouvelle approche pour la spécification et la conception de transformations de modèles appelée MoTrans-BDI, qui exploite le système multi-agents évolutif (EMAS) pour simuler l'expertise des concepteurs pour la transformation de modèles. Notre approche est basée sur le modèle d'agent Belief-Desire-Intention (BDI) et le Contract Net Protocol où les croyances des agents se nourrissent d'une série d'exemples de transformation. L'accent mis sur l'utilisation du modèle spécifique est la possibilité de produire un modèle cible qui peut être composé de pièces provenant de conceptions de différents experts. Nous évaluons expérimentalement MoTrans-BDI sur douze problèmes de transformation de modèle UML2REL faits à la main. Tous les types d'agents sont capables de produire des modèles cibles parfaits par rapport aux modèles cibles produits en collaboration par des experts humains. Une application de MoTrans-BDI dans le cas de la transformation UML2REL est effectuée pour souligner et mettre en évidence les étapes du processus de transformation.

Mots-clés : Transformation de modèle par l'exemple, ingénierie pilotée par les modèles, modèle croyance-désir-intention, système multi-agents, protocole Contract Net.

ملخص

أجمع الباحثون على أن نظام إدارة قواعد البيانات (Database Management System) هو واحد من أكثر المرشحين الوعدين ليكون العمود الفقري للخدمات المعلوماتية الوعية بالجودة. عادة، يتم قياس هذه الجودة من خلال مقاييس ذاتية والتي يمكن تفسيرها على أنها الدرجة التي تمتلك بها أنظمة قواعد البيانات وظيفة معينة و التي توفر على جودتها.

إن جودة الخدمة (Quality of Service) تتعلق بجميع مراحل دورة حياة تصميم تطبيقات قواعد البيانات ، مع اهتمام خاص بمرحلة التصميم الفيزيائي - التي تعتبر بمثابة مسار لمراحل أخرى وتفاعل بشدة مع معالجة الإستعلامات. يتم ضمان عملية تحقيق جودة الخدمة في سياق التصميم المادي لقواعد البيانات من خلال مقاييس و التي يمكن اعتبارها كدالة بحيث (1) تتنمي معلمات الإدخال الخاصة بها إلى تطبيق قاعدة البيانات (بما في ذلك مخططها واستعلاماتها)، و إلى نظام قاعدة البيانات المستضيف لهذا القاعدة بالإضافة إلى منصة النشر الخاصة بها، و (2) يكون ناتجها عبارة عن قيمة عددية واحدة. يتم التعبير عن هذه المقاييس عادة من خلال نماذج التكلفة التحليلية، إن تطوير هذه النماذج يستغرق وقتاً طويلاً ويطلب المعايرة لعكس تطور تقنية قواعد البيانات (التي تشير إلى كل من البرامج والأجهزة). في مواجهة هذا الوضع، فإن وجود أدوات مخصصة لتطوير، إستغلال ومعايرة نماذج التكلفة أصبح ضرورة للباحثين والطلاب. في هذا البحث، نقترح أولاً لغة نطاق محددة لتطوير نماذج التكلفة للتصميم المادي لقواعد البيانات مع تمكين جودة الخدمة، يستلزم تطوير هذه اللغة تحديد مختلف مدخلات نموذج التكلفة بفضل تقنيات النمذجة الفوقيّة. وثانياً ، لزيادة إعادة استخدام هذه النماذج ، نقوم بإقتراح مستودع مستديم لحفظ نماذج التكلفة التي تم تطويرها حديثاً باستخدام تطبيق ويب. وأخيراً ، يتم توفير مجموعة أدوات، تسمى MetricStore، بما في ذلك الوظائف المختلفة المتعلقة بإدارة نماذج التكلفة.

الكلمات المفتاحية: التصميم الفيزيائي، نماذج التكلفة ، مستودع، لغة خاصة بالنطاق ، جودة الخدمة.

Abstract

Due to the growing interest in model-driven system development techniques, the efficient design of automated model transformations between heterogeneous models has become a major challenge in software development. While a number of specialized languages have been proposed, aiming at specifying model transformations, there is currently no matured foundation for specifying transformations between such models that are based on designers/experts collaboration in order to propose a solution that satisfies users' requirements and application constraints. This transformation process is a complex task and must emulate how designers and experts with different perspectives behave and reflect about model transformations. In this paper, we propose a framework based on a novel approach for the specification and design of model transformations called MoTrans-BDI, which leverages Evolutionary Multi-Agent System (EMAS) to simulate designers' expertise for the transformation of models. Our approach is based on the Belief-Desire-Intention (BDI) agent model and the Contract Net Protocol where agents' beliefs feed from a series of transformation examples. The emphasis of using the specific model is the opportunity to produce a target model that may be composed of parts from different experts' designs. We experimentally evaluate MoTrans-BDI on twelve handmade UML2REL model transformation problems. All types of agents are able to produce perfect target models compared to human experts' collaboratively-produced target models. An application of MoTrans-BDI in the case of UML2REL transformation is performed to stress and highlight the transformation process steps.

Keywords : Model Transformation By Example, Model-Driven Engineering, Belief-Desire-Intention Model, Multi-Agent System, Contract Net Protocol.

Remerciements

C'est avec grand plaisir que je réserve cette page, en signe de gratitude et de reconnaissance à tous ceux qui m'ont aidé à la réalisation de ce travail.

Je remercie, tout d'abord, **Ladjel BELLATRECHE** pour sa rigueur et la pertinence de ses jugements qui ont été très constructifs et m'ont permis de faire ce travail. Je lui suis également reconnaissante pour sa contribution à l'amélioration judicieuse de la qualité de ce document.

Je remercie **Kamel BOUKHALFA** pour son co-encadrement minutieux et ses bonnes qualités pédagogiques et scientifiques. Ses conseils avisés ont été précieux et sa gentillesse mégalable.

Je remercie **x** pour m'avoir fait l'honneur d'être président du jury,

Je remercie cordialement **x** et **x** d'avoir accepté d'être rapporteurs de cette thèse. Je souhaite adresser également mes remerciements à **x**, **x** et **x** d'avoir fait l'insigne honneur d'accepter d'examiner mon travail ;

Un grand merci à **Yassine OUHAMMOU**, pour sa contribution dans tous mes papiers de recherche, pour avoir consacré du temps à la lecture de mes articles et pour sa patience et ces encouragements.

Ce travail de thèse a été réalisé dans le cadre de stage PNE dans LIAS. Je remercie leur directeur **Emmanuel GROLLEAU** ainsi que tous les chercheurs et l'ensemble du personnel du laboratoire pour m'avoir fait profiter de leur expérience et de leur savoir.

À tous les membres d'Université de Tiaret et du laboratoire LIAS À tous les membres du laboratoire LIAS, en particulier : Madjid, Amine, Selma, Lahcène, Zouhir, ainsi que Selma Khouri, Sabrina, Dalila et Ibrahim avec lesquels j'ai partagé des moments inoubliables.

Je voudrais exprimer à mes proches toute ma gratitude : mes très chers parents, mes frères et mes sœurs et mes belles nièces.

Sans leur soutien, leur confiance et leurs encouragements, je n'y serais jamais arrivé.

À mes amis et collègues sans exception, pour leur présence, leur respect et leur gentillesse.

Et enfin, merci à tous ceux qui ont participé de près ou de loin à l'aboutissement de ce travail.

À tous ceux qui me sont chers :
Ma mère, mon Père
Mes frères, Mes soeurs.



Table des matières

Liste des figures	xvii
Liste des tableaux	xix
Partie I Introduction Générale	1
Chapitre 1 Introduction Générale	3
1.1 Contexte et Motivation	4
1.2 Problem statement	4
1.3 Contribution	5
1.4 Paper Outline	6
Partie II État de l'art	7
Chapitre 2 Spécification des règles de transformation des modèles	9
2.1 Introduction	10
2.2 Ingénierie dirigée par les modèles	11
2.2.1 Introduction	11
2.2.1.1 Introduction	11
2.3 Transformation de modèle	11
2.4 Spécification explite des règles de transformation	11
2.4.1 Spécification explite des règles de transformation	11
2.5 Introduction	11
2.6 Introduction	11
2.7 Introduction	11
2.8 Conclusion	11
2.9 Concepts de base	11

Table des matières

2.9.1	Ingénierie Dirigée par les modèles	12
2.9.2	Modèle	12
2.9.3	Métamodèle	12
2.9.4	Transformations des modèles	13
2.9.5	Règles de transformation	13
2.9.6	xDSML	14
2.10	Les langages de transformation des modèles.	15
2.10.1	Le langage de métamodélisation MOF	16
2.10.2	Kermeta :	16
2.10.3	Ecore	16
2.10.4	ATOM	16
2.10.5	AGG	16
2.10.5.1	Les framework de transformation des modèles.	18
2.10.6	Quelques approches de transformation de modèle en détail	19
2.10.7	La classifications des modèles de transformation	20
2.10.7.1	Topologie de transformation	22
2.10.7.2	Exactitude de la transformation du modèle	23
2.11	Conclusion	23
Chapitre 3 Les approches de Transformation des modèles par l'exemple		25
3.1	Introduction	26
3.2	Pourquoi L'exemple	26
3.3	Formalisation de modèle de transformation par l'exemple	27
3.4	Classification des approches MTBE	27
3.4.1	Approche explicite	27
3.4.1.1	Travaux de xxxxx et a	27
3.4.1.2	Travaux de xxxxx et a	27
3.4.1.3	Travaux de xxxxx et a	27
3.4.1.4	Travaux de xxxxx et a	27
3.4.1.5	Travaux de xxxxx et a	27
3.4.1.6	Travaux de xxxxx et a	27
3.4.1.7	Travaux de xxxxx et a	27
3.4.1.8	Travaux de xxxxx et a	27
3.4.1.9	Travaux de xxxxx et a	27
3.4.2	Approche implicite	27

3.4.2.1	Travaux de xxxxx et a	27
3.4.2.2	Travaux de xxxxx et a	27
3.4.2.3	Travaux de xxxxx et a	27
3.4.2.4	Travaux de xxxxx et a	27
3.4.2.5	Travaux de xxxxx et a	27
3.4.2.6	Travaux de xxxxx et a	27
3.4.2.7	Travaux de xxxxx et a	27
3.4.2.8	Travaux de xxxxx et a	27
3.5	Bilan des travaux antérieurs sur les travaux MTBE	27
3.5.1	Le problème de déduction de la régles de la transformation	27
3.5.1.1	Problèmes adjacents à la déduction de ka règles	27
3.5.2	Incertitudes liées à la mmaturité de domaine de connaissance	27
3.5.3	Méthodes de résolution	27
3.5.3.1	Méthodes basées recherche /optmisation	27
3.5.3.2	Méthodes basées apprentissage	27
3.5.4	Synthèse	27
3.5.5	Conclusion	27
3.6	Concepts de base de TMPE	28
3.6.1	Motivation	28
3.6.2	Pourquoi l'exemple ?	29
3.6.3	Derivation de la règle de transformation	29
3.6.4	Les approches de Transformation de modèle a base d'exemples	32
3.6.4.1	Travaux de Varro	32
3.6.4.2	Travaux de Wimmer	32
3.6.4.3	Travaux de Garcia-Magarino	32
3.6.4.4	Travaux de Kessentini	33
3.6.4.5	Travaux de Dolques	33
3.6.4.6	Travaux de Baki	34
3.6.4.7	Travaux de Faunes	34
3.6.5	Carte conceptuelle des approches MTBE (Approches Timeline)	35
3.6.6	Synthèse	37
3.7	Conclusion	37

Partie III Contributions	39
Chapitre 4 Cadre fondé sur les agents BDI pour la transformation collaborative des modèles à base de l'exemple	41
4.1 Introduction	42
4.2 MoTrans-BDI Framework and its Theoretical Foundations	42
4.2.1 The MoTrans-BDI System Overview	42
4.2.2 Theoretical Foundation of MoTrans-BDI Framework	43
4.2.3 Conceptual Organization of MoTrans-BDI Framework	44
4.2.4 The MoTrans-BDI Process	46
4.2.4.1 Sending a call for proposal	46
4.2.4.2 Model transformation processing and sending of proposals . . .	46
4.2.4.3 Refinement of the proposed target models	51
4.2.4.4 Agent Learning	51
4.3 Conclusion	52
Chapitre 5 Adaptation de l'Algorithme Génétique pour le problème de reproduction des exemples de transformation des modèles	53
5.1 Introduction	54
Chapitre 6 Adaptation de l'Algorithme Génétique pour la Transformation des Modèles	55
6.1 Introduction	55
6.2 Problèmes de transformation des modèles	55
6.2.1 Formalisation du problème	55
6.2.2 Démarche de transformation	56
6.2.3 Modèle de coût	59
6.2.4 Modèle de coût pour la transformation	59
6.2.5 Fonction objectif pour l'algorithme génétique	60
6.2.5.1 Formulation d'une fonction objectif	60
6.2.5.2 Fonction objectif pour la fragmentation	61
6.2.6 Mécanisme de codage d'un schéma de transformation	62
6.2.6.1 Génération d'un modèle de transformation à partir d'un codage	62
6.2.6.2 Fonction Fitness	63
6.2.6.3 Génération de la population initiale	63
6.2.6.4 Sélection	64
6.2.6.5 Croisement	64

6.2.6.6	Mutation	65
6.3	conclusion	66
Chapitre 7 Outil Suppport de notre cadre de transformation MTBE		67
7.1	Introduction	68
7.2	MoTrans-BDI Implementation and Evaluation	68
7.2.1	Implementation	68
7.2.2	Evaluation of MoTrans-BDI	70
7.2.2.1	Creating the dataset	70
7.2.2.2	Evaluation's Data Collection	70
7.2.2.3	Evaluation Metrics	71
7.2.2.4	Experiment strategy and procedure	72
7.2.2.5	BTAs Model Transformation Correctness Testing	72
7.2.2.6	MoTrans-BDI Accuracy Testing	73
7.2.2.7	Collective design Vs Individual design	74
7.2.3	Evaluation of MoTrans-BDI proposal's impact against Performance Measurement	74
7.3	Threats to validity	76
Chapitre 8 Conclusion et Perspectives		83
8.1	Introduction	84
8.2	Conclusion	84
8.3	Perspectives	84
Partie IV Conclusion et perspectives		85
Chapitre 9 Conclusion générale et Perspectives		87
9.1	Conclusion	88
9.1.1	État de l'art	88
9.1.2	Contribution 1	88
9.1.3	Contribution 2	88
9.1.4	Preuve de concept	88
9.2	Perspectives	88
9.2.1	Ajout des nouvelles fonctionnalités	88
9.2.2	Qualité de MdCs partagés	88
9.2.3	Explorer l'analyse des relations entre les MdCs	88

Table des matières

9.2.4	Rapide prototype des MdCs	88
9.2.5	Utilisation de notre framework en mode pédagogique d'enseignement . . .	88
Partie V	Annexes	89
Annexe A	Questionnaire Académique en ligne de l'évaluation de l'acceptabilité de MetricStore	91
A.1	Questionnaire Académique	91
Annexe B	Lexique	93
Bibliographie		95





Liste des figures

2.1	L'architecture MOF	12
2.2	Exemple d'un modèle UML.	12
2.3	Concepts de base de transformation de modèles.	13
2.4	Les trois types des syntaxes : abstraire, concrète et sémantique	13
2.5	Processus de transformation des modèles [39]	13
2.6	Métamodèle d'une règle de transformation.	14
2.7	Example d'une règle de transformation : cas de jeux pacman	14
2.8	Petri net xDSML	15
2.9	Les outils de transformation des modèles.	18
2.10	Vue d'ensemble de l'outil Xtext.	19
2.11	Approches de transformation de modèle.	22
3.1	Principe de transformation de modèle par l'exemple [20].	29
3.2	Exemple d'un exemple de paire source-cible avec trois traces identifiées	30
3.3	Illustration de la transformation	30
3.4	Classification des approches de transformation des modèles.	31
3.5	Méta-modèles simplifiés de UML (à gauche) et Entité-Relation (à droite) utilisés dans wimmer et al 2007[15].	33
3.6	Carte conceptuelle des approches MTBE (Approches Timeline)	35
4.1	Overview of MoTrans-BDI Framework	42
4.2	Conceptual Organization of MoTrans-BDI Framework	45
4.3	Excerpt of our design language of model transformation heuristics.	45
4.4	The MoTrans-BDI model transformation process.	46
4.5	Example CDS_{y3} corresponding to a fragment of TM_{y3}	48
4.6	Example of similarity coefficients for Binary Data	49

Liste des figures

4.7	Example of crossover of two parents (chromosomes) corresponding to two TMs	49
4.8	The BTA model transformation process	51
4.9	Refinement scenarios illustration.	51
6.1	Démarche des modèles de transformation	56
6.2	Figure 2 :Schéma d'un modèle source	57
6.3	Figure 3 :Codage d'un schéma de Transformation	58
6.4	Figure 9 :Exemple de croisement	65
6.5	Figure 10 :Exemple de mutation	65
7.1	MoTrans-BDI main GUI for CFP and Transformation services.	78
7.2	MoTrans-BDI main GUI for Refinement and Configuration services.	79
7.3	Model Transformation Data gathering.	79
7.4	BTAs Model Transformation Correctness Testing.	80
7.5	TMs Similarity Degree with and without refinement.	80
7.6	MA's TMs similarity Vs BTAs' TMs similarity	80
7.7	The initial TM corresponding to the SSB's SM	81
7.8	Cumulative queries performance cost corresponding to the initial model, the BTAs, and the MA	81





Liste des tableaux

2.1	Approches de langages de la transformation des modèles.	22
3.1	Caractéristiques des approches TMPE actuelles.	35
7.1	MoTrans-BDI Response time costs	73
7.2	Summary of <i>MoTrans-BDI</i> comparison metric values	74
7.3	SSB benchmark queries response time estimations.	76

Première partie

Introduction Générale



Introduction Générale

« *There is nothing more difficult to take in hand, more perilous to conduct, or more uncertain in its success, than to take the lead in the introduction of a new order of things.* »
— Niccolo Machiavelli. 1469 -1527

Sommaire

1.1	Contexte et Motivation	4
1.2	Problem statement	4
1.3	Contribution	5
1.4	Paper Outline	6

1.1 Contexte et Motivation

Model transformation is a cornerstone of the model-driven engineering (MDE) paradigm [6], [7]. A transformation takes as input a source model (SM) and produces a target model (TM). A popular example of transformation is UML2REL that transforms a UML class diagram into a relational schema for which elements in the target model are created using information from the source model. Writing transformations is a resource and time-consuming task that requires knowledge of both domains, knowledge of the semantics of the transformation language, and in-depth expertise on the quality of service the result must satisfy (e.g. *performance, storage, usability, security*) [25, 26].

In general a model transformation is written manually using a transformation language. Alternatively, it can be derived with a certain degree of automation from examples of inputs-outputs, i.e., *model transformation by the example* (MTBE) [3, 34, 35, 5]. The notion of MTBE, initially proposed by Varro [3], is inspired by other by-example approaches like Programming By-Example or Querying By-Example. This approach of model transformation aims at learning automatically the coveted transformations from a set of source and target model pairs provided as examples. The transformation process must reflect some human behavior aspects, namely, the collaboration and different design views to collect all the needs, to resolve proposal conflicts, and to satisfy user requirements.

Over the last two decades, a large amount of ongoing research effort has been invested in model transformation [4, 38, 23, 17, 8], and several approaches and tools have been proposed specifically for MTBE [24, 28, 1]. Most of these approaches are based on a single repository of examples without an explicit structure reflecting the diversity of sources of these examples, and in particular, how they convey experts' views of design. However, we believe that the idea of proposing different experts' design alternatives for examples selection may enhance the quality of transformations and cover all their possible design needs. This idea is inspired by the concept of co-design in companies where different experts collaborate to integrate and exploit requirements that are based on different perspectives in a context of design sharing [36].

1.2 Problem statement

As stated, a large number of tools supporting model transformation have been developed. For MTBE, each of these tools is based on a single set of source and target model pairs provided as examples that reflect the expert or researcher's knowledge and differ significantly in their capabilities and limitations, making it difficult to select which pair best covers any given part of the model to transform. Additionally, in particular fields such as Robotics and Autonomous Systems [13], the domain knowledge is not mature enough due to the lack of consensus between a multiple point-of-view. This gap related to the absence of consensus in model transformation is a strong argument to adopt a new MTBE approach. To facilitate the effective use of model transformation in practice and consolidate existing research and development results, multiple sources of examples can be combined to consider explicitly the diversity of knowledge and experience that may lead to the transformation building. Such a combination can be achieved through a high-level approach that allows the analysis and refinement of examples' contributions to select the best ones to consider based on some heuristics expressing users' requirements and application constraints. Such an approach must support a distributed decision-making with the capabilities of

finding the best trade-offs by evaluating the alternatives, relaxing some requirements and resolving conflicts.

1.3 Contribution

The main contribution of this paper is to lay out a by-example based approach that is inspired from Contract Net Protocol where a subcontractor agent initializes the protocol by proposing a task to several agents : the contractors [14]. The contractors can send either a proposal if they are interested or a reject if they are not. This proposal is provided with all the elements required by the subcontractor to make its choice. More concretely, MoTrans-BDI is based on the Belief-Desire-Intention (BDI) agent model coupled with Contract Net Protocol architecture. A main agent (MA) is responsible of sending the source model, or a part of it, to several intelligent BDI agents, called the BDI-Transformer-Agents (BTA), that are playing the role of contractors. Each transformer agent sends back a proposal that represents the target model, or a part of it, to the MA. Each agent may possibly elaborate its proposal using its own strategy that reflects an expert's design view. Although, in this paper, all agents use a unique strategy that is based on a set of transformation examples. Then, the MA analyses all proposals to select the best transformation solution. This solution is selected based on the goal of transformation that expresses user's needs and application constraints to achieve a certain quality of service (QoS) [22].

The derivation of a solution by each BTA is treated as an optimization problem, and solved using an evolutionary algorithm. The combination of evolutionary algorithms with multi-agent systems is known as Evolutionary Multi-Agent Systems (EMAS) [12]. In such a system, the agents have the ability, among others, to communicate with other agents, to learn, and to make autonomous decisions. Moreover, the task of model transformation is distributed through several experts according to their design views. The idea of using EMAS is motivated by the fact that the model architecture is more generic insofar as the beliefs of each BTA as well as its MTBE algorithm can change according to the model transformation goal and the way the expert transforms models respectively.

In order to validate MoTrans-BDI, we implemented a prototype tool for transforming a UML class diagram into a relational schema. We carried out a series of experiments on UML2REL cases to show the feasibility and significance of MoTrans-BDI taking into account some QoS expressed in a form of heuristics.

The contributions of this work are summarized as follows :

1. Combination of the BDI paradigm with Contract Net Protocol architecture to handle the model transformation by example process.
2. Distribution of the experts' decision-making on model transformation, and derivation of the different proposals by explicitly considering QoS heuristics, following an evolutionary algorithm.
3. Implementation of a prototype tool to support the proposed approach.

1.4 Paper Outline

The rest of our paper is structured as follows : Section ?? presents a motivation example and gives an overview of the existing work on Model Transformation By Example. Section [Section 4.2](#) details the framework components. In section [Section 7.2](#), we present an implementation of the approach and its evaluation. [Section ??](#) clarifies the threat to validity. Finally, we conclude our paper by highlighting the main contributions and by giving some future work perspectives in section [Section 8.3](#).

Deuxième partie

État de l'art

Spécification des règles de transformation des modèles



« Measurement is the first step that leads to control and eventually to improvement. »

— H. James Harrington

Sommaire

2.1	Introduction	10
2.2	Ingénierie dirigée par les modèles	11
2.2.1	Introduction	11
2.3	Transformation de modèle	11
2.4	Spécification explite des règles de transformation	11
2.4.1	Spécification explite des règles de transformation	11
2.5	Introduction	11
2.6	Introduction	11
2.7	Introduction	11
2.8	Conclusion	11
2.9	Concepts de base	11
2.9.1	Ingénierie Dirigée par les modèles	12
2.9.2	Modèle	12
2.9.3	Métamodèle	12
2.9.4	Transformations des modèles	13
2.9.5	Règles de transformation	13
2.9.6	xDSML	14
2.10	Les langages de transformation des modèles.	15
2.10.1	Le langage de métamodélisation MOF	16
2.10.2	Kermeta	16
2.10.3	Ecore	16
2.10.4	ATOM	16
2.10.5	AGG	16
2.10.6	Quelques approches de transformation de modèle en détail	19
2.10.7	La classifications des modèles de transformation	20
2.11	Conclusion	23

2.1 Introduction

L'ingénierie dirigée par les modèles (MDE) [27] est une discipline relativement nouvelle, qui à abstraire le développement logiciel à partir de la technologie de mise en œuvre et de réduire sa complexité en déplaçant l'attention de la phase de codage vers la phase de modélisation. Dans cette vue centrée sur le modèle, les transformations de modèles deviennent des citoyens de première classe : les modèles peuvent être vus comme des représentations du logiciel tandis que les transformations de modèles peuvent être considérées comme le collage mécanique entre les modèles [39]. Ainsi, à partir d'un modèle et aux moyens de transformations de modèles, il est possible d'obtenir automatiquement une variété d'autres artefacts. En particulier, les transformations de modèles peuvent être exploitées pour une grande variété de tâches non limitées à la génération au code source , telles que l'analyse, les tests, la vérification de la cohérence du modèle et la synchronisation du modèle. L'adoption du paradigme MDE a conduit à une croissance rapide de la discipline, qui est continuellement confrontée à des défis de plus en plus complexes. Transformation de modèle bidirectionnelle (BX) est un défi qui se pose dès que des modifications sur le modèle cible sont autorisées, étant donné que ces modifications doivent être mappées à la source afin d'éviter leur perte en raison de l'écrasement de cible [30] . Une première tentative de développement d'un langage BX standard est représentée par Relations QVT (QVT-R), qui est incluse dans la transformation de vue de requête (QVT) [9] par le groupe de gestion d'objets (OMG). Bien que affecté par plusieurs faiblesses, comme discuté dans ?????, il a clairement établi la pertinence de BX dans MDE.

2.2 Ingénierie dirigée par les modèles

2.2.1 Introduction

2.2.1.1 Introduction

2.3 Transformation de modèle

2.4 Spécification explite des règles de transformation

2.4.1 Spécification explite des règles de transformation

2.5 Introduction

2.6 Introduction

2.7 Introduction

2.8 Conclusion

selves, which are in general neither injective [22] nor total [14] : each time a change occurs, more than one reverse function may be available ; moreover, in general the transformation only involves a subset of elements of the source and target models. A number of approaches and languages have been proposed due to the intrinsic complexity of bidirectionality. Each one of those languages is characterized by a set of specific properties pertaining to a particular applicative domain [5, 23]. Despite the large number of available model transformation approaches and the common understanding about the importance of bidirectionality, there is a lack of standardisation : most of the BX issues are still not properly investigated ; more important, there is not a clear understanding on which features a BX language should provide to overcome specific bidirectionality issues.

2.9 Concepts de base

Cette section vise à donner une vue simplifiée des concepts clés et de la terminologie impliqués dans l'IDM, en répondant à des questions courantes comme : Qu'est-ce qu'un modèle, un métamodèle et un métaméta-modèle ? Quelle est la relation entre ces concepts ? Quelles sont les facettes clés d'un langage de modélisation et d'une méthode de modélisation, Et qu'est-ce qu'un point de vue ?

2.9.1 Ingénierie Dirigée par les modèles

L'IDM est une approche de développement mettant à disposition des outils, des concepts et des langages afin de simplifier et de mieux maîtriser le processus de développement de systèmes qui ne cessent de croître en complexité. D'autre part, elle permet aussi d'augmenter la productivité, la qualité, la réutilisabilité et l'évolution de ces systèmes.



chapitre2/chap2Fig/fig1.png

FIGURE 2.1 – L'architecture MOF.

2.9.2 Modèle

Nous avons trouvé dans la littérature plusieurs définitions du terme modèle.

Définition 1

Un modèle est une abstraction d'un système construit dans un but précis.

Définition 2

Un modèle est une abstraction qui contient un ensemble limité d'informations sur un système.

Définition 3

Un modèle est construit dans un but précis et les informations qu'il contient doivent être pertinentes et adaptées au usage spécifique auquel le modèle sera destiné (Muller, 2006). La figure Figure 2.2 illustre un exemple d'un modèle UML



chapitre2/chap2Fig/fig2.png

FIGURE 2.2 – Exemple d'un modèle UML.

2.9.3 Métamodèle

Définition 4

Selon Kühne [kuhne2006matters], le métamodèle définit les types linguistiques et leurs relations. Un métamodèle est défini à l'aide d'un langage de métamodélisation, tel que UML.

Définition 5

En pratique, nous définissons des métamodèles à l'aide de diagrammes de classes UML, où les classes représentent les concepts de type, les attributs représentent leurs propriétés et les associations représentent leurs rapports, la figure Figure 2.3 illustre les concepts de base de transformation de modèles.



FIGURE 2.3 – Concepts de base de transformation de modèles.

Un métamodèle est un modèle de langage qui décrit ses concepts et ses propriétés, sa syntaxe textuelle et/ou graphique et sa sémantique. Comme illustrée dans la figure Figure 2.4 Un DSL peut s'exprimer en trois types de syntaxe :

- **Syntaxe abstraite** : La syntaxe abstraite (AS) d'un langage de modélisation exprime, de manière structurelle, l'ensemble de ses concepts et leurs relations.
- **Syntaxe concrète** : Les syntaxes concrètes (CS) d'un langage fournissent à l'utilisateur un ou plusieurs formalismes, graphiques et/ou textuels, pour manipuler les concepts de la syntaxe abstraite et ainsi en créer des « instances ». Le modèle ainsi obtenu sera conforme à la structure définie par la syntaxe abstraite.
- **Syntaxe sémantique** : Définir la sémantique d'un langage ce qui revient à définir le domaine sémantique et le mapping entre la syntaxe abstraite et le domaine sémantique (AS <=> SD). Le domaine sémantique définit l'ensemble des états atteignables par le système, et le mapping permet d'associer ces états aux éléments de la syntaxe abstraite.



FIGURE 2.4 – Les trois types des syntaxes : abstraire, concrète et sémantique

2.9.4 Transformations des modèles

Les auteurs fournissent les définitions suivantes de MT :

Définition 6

Les transformations de modèles peuvent être considérées comme le cœur du MDE : sans une maturité technologique de la transformation, la qualité et l'efficacité d'IDM ne pouvaient pas être accordé.

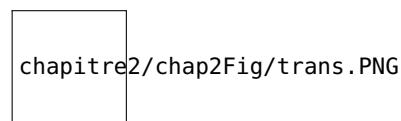


FIGURE 2.5 – Processus de transformation des modèles [39]

2.9.5 Règles de transformation

Une règle de transformation R est définie par un tuple $R = (NAC, LHS, RHS)$, où LHS et RHS sont les graphes typés appelés graphes de gauche et graphe de droite de la règle, et (NAC)représente un

ensemble (potentiellement vide) de graphes typés appelés les conditions d’application négatives. Deux exemples de règles ont été définis comme le montre la figure ?? Dans la règle 1, les actions entrantes sur un état X sont repliées dans l’état en tant qu’action d’entrée. Si deux fronts entrants d’un état X ont une action définie, ils sont supprimés des bords et X aura l’action d’entrée A. Le NAC précise que X ne peut pas déjà avoir une action d’entrée pour que la règle soit applicable.

Pour résumé :

– **LHS,NAC :**

Sont les règles de la précondition qui doivent être vérifier avant l’application de la règle (le modèle cible).

– **RHS (Right-Hand-Side) :**

Sont les règles de la postcondition qui doivent être vérifiées dans le modèle cible .



FIGURE 2.6 – Métamodèle d’une règle de transformation.



FIGURE 2.7 – Exemple d’une règle de transformation : cas de jeux pacman

2.9.6 xDSML

Définition 7

Un xDSML est défini par :

- Une syntaxe abstraite, c'est-à-dire un métamodèle.
- Sémantique opérationnelle, composée de :
 - Un métamodèle d'exécution, qui définit l'exécution état des modèles exécutés en étendant la syntaxe abstraite avec de nouvelles propriétés et métaclasses à l'aide de la fusion de packages ou de tout mécanisme similaire.
 - Une transformation d'initialisation, c'est-à-dire une transformation de modèle exogène qui transforme un modèle conforme à la syntaxe abstraite en un modèle conforme au métamodèle d'exécution, alors qu'au moins préserver le contenu du modèle d'entrée.
 - Une transformation d'exécution, c'est-à-dire une transformation de modèle sur place qui modifie un modèle conforme au métamodèle d'exécution en modifiant les valeurs des champs dynamiques et en créant/détruisant des instances de métaclasses introduites dans le métamodèle d'exécution. Le sous-ensemble de règles de transformation qui sont considérées comme observables sont appelées règles par étapes .

La Figure [Figure 2.8](#) montre un exemple de réseau de Petri simple xDSMLL. En haut à gauche, sa syntaxe abstraite est représentée avec trois métaclasses Net, Place et Transition. À côté de la syntaxe abstraite, le métamodèle d'exécution est affiché. Il étend la métaclassse Place en utilisant la fusion de packages avec de nouveaux jetons de propriété dynamiques. La fonction d'initialisation (non représentée) transforme chaque objet d'origine (c'est-à-dire un objet Place sans champ jetons) en un objet exécutable (c'est-à-dire un objet Place avec un champ jetons) tel que défini dans le métamodèle d'exécution. Il initialise également chaque champ de jetons avec la valeur de initialTokens. En bas, les descriptions des règles définies dans la sémantique opérationnelle sont représentées. Lorsqu'elles sont appelées, ces règles peuvent modifier les champs de jetons des différents objets Place, run étant le point d'entrée de la transformation.

L'étiquette @Step est utilisée pour indiquer quelles règles de transformation sont des règles d'étape. En étiquetant run et fire avec @Step, nous spécifions que le modèle n'est observable qu'avant et après l'application de ces règles, c'est-à-dire avant et après l'exécution (puisque run est le point d'entrée) et avant et après le déclenchement des transitions. les jetons des places d'entrée sans ajouter de jetons aux places de sortie, mais seulement après le déclenchement complet d'une transition.

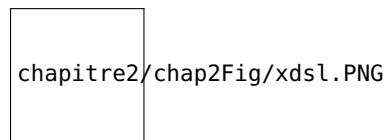


FIGURE 2.8 – Petri net xDSML

Définition 8

Une trace d'exécution est une séquence d'exécution les états et les étapes d'exécution (petites et grandes étapes) responsables des changements d'état.

Définition 9

Une étape d'exécution est l'application d'une étape régner. Une étape d'exécution qui n'est pas composée d'autres étapes est appelé une petite étape, tandis qu'une étape d'exécution composée de plusieurs étapes s'appellent une grande étape.

Définition 10

Un état d'exécution est l'ensemble des valeurs de tous les champs dynamiques d'un modèle à un certain moment de l'exécution. L'état d'exécution d'un modèle est modifié par l'application des règles de la transformation d'exécution.

2.10 Les langages de transformation des modèles.

Dans cette section, nous allons présenter quelques langages de transformation des modèles.

2.10.1 Le langage de métamodélisation MOF

Le langage MOF (Meta Object Facility) se situe au sommet dans l'architecture de l'OMG à quatre niveaux. C'est un méta-formalisme, pour établir des langages de modélisation permettant eux-mêmes d'exprimer des modèles. Dans sa version 2.0, le méta-métamodèle MOF est constitué de deux parties : EMOF (Essential MOF), pour l'élaboration des métamodèles sans association, et CMOF (Complete MOF) pour l'élaboration des métamodèles avec associations. Il faut souligner que MOF s'auto-décrit pour pouvoir limiter l'architecture de l'OMG à quatre niveaux. Aussi, le langage MOF emprunte la syntaxe du langage de modélisation UML .

2.10.2 Kermeta :

Il est défini comme un langage de métamodélisation exécutable ou de métaprogrammation objet, ce qui signifie qu'il permet de décrire des métamodèles dont les modèles sont exécutables. Kermeta est basé sur le langage de métamodélisation EMOF et il est intégré à l'IDE Eclipse sous la forme d'un plugin. Il a été conçu comme un tissage entre un métamodèle comportemental et EMOF. Le métamodèle de Kermeta est donc composé de deux packages : Core et Behavior. Le premier correspond à EMOF et le second est une hiérarchie de métaclasses représentant des expressions impératives.

2.10.3 Ecore

Le langage Ecore a été présenté par la fondation Eclipse [[baldassari2014ombre](#)] dans le cadre du projet EMF (Eclipse Modeling Framework) [[budinsky2004eclipse](#)]. Le projet EMF propose un framework pour le développement des applications basées sur l'ingénierie dirigée par les modèles. Ce framework permet de générer automatiquement des interfaces Java à partir des métamodèles Ecore

2.10.4 ATOM

ATOM "Domain Specific Visual Languages" [est un outil pour la conception de Domain Specific Visual Languages (DSML). Il permet de définir la syntaxe abstraite et concrète d'un langage visuel au moyen de la métamodélisation et d'exprimer la manipulation du modèle en utilisant la transformation de graphe. Avec les informations de métamodèle, ATOM génère un environnement de modélisation personnalisé pour le langage décrit. Récemment, ATOM a été étendu avec des fonctionnalités pour générer des environnements avec des vues multiples de langages visuels (tels que UML) et Triple Graph Grammar (TGG) . Ce dernier est utile pour exprimer l'évolution de deux modèles différents, liés par un modèle intermédiaire.

2.10.5 AGG

AGG " Algebraic Graph Grammar" est un environnement à usage général pour le développement des systèmes de transformation de graphes attribués. Il est basé sur l'approche algébrique pour la transformation de graphes. Il vise à la spécification et le prototypage rapide d'applications complexes tel que le graphe de données structurées. Puisque la transformation de graphe peut être appliquée à des

niveaux d'abstraction très différents, elle peut être attribuée ou non par des calculs simples ou par des processus complexes, selon le niveau d'abstraction. En raison de sa base formelle, AGG offre un support de validation, de vérification de la cohérence des graphes en fonction des contraintes graphiques

(Voir la [Section 2.10.5](#))

2.10.5.0.1 ATL :

Le langage ATL (ATLAS Transformation Language) est défini dans le contexte des langages de transformation par métamodèles. L'opération de transformation est dirigée par un programme ATL nommé mma2mmbl.atl. Ce programme transforme le modèle source Ma en un modèle cible Mb. Il est lui-même un modèle. Les modèles Ma, Mb et le programme sont conformes aux métamodèles MMA, MMb et ATL respectivement. Ces métamodèles sont conformes au métaméta-modèle MOF.

2.10.5.0.2 Acceleo :

Acceleo est un générateur de code source de la fondation Eclipse permettant de mettre en œuvre l'approche MDA (Model driven architecture) pour réaliser des applications à partir de modèles basés sur EMF. Il s'agit d'une implémentation de la norme MOF Models to Text (MOFM2T) de l'Object Management Group (OMG) pour les transformations de modèles à texte.

2.10.5.0.3 xtend :

Langage de programmation Xtend, un langage de type Java complet étroitement intégré à Java. Xtend a une syntaxe plus concise que Java et fournit des fonctionnalités supplémentaires telles que l'inférence de type, les méthodes d'extension et les expressions lambda, sans parler des expressions de modèle multilignes (qui sont utiles lors de l'écriture de générateurs de code). Tous les aspects d'un DSL implémenté dans Xtext peuvent être implémentés dans Xtend au lieu de Java, car il est plus facile à utiliser et permet d'écrire du code plus lisible. Étant donné que Xtend est complètement interopérable avec Java, vous pouvez réutiliser toutes les bibliothèques Java ; de plus, tous les JDT Eclipse (Java Development Tools) fonctionneront avec Xtend¹.

2.10.5.0.4 Epsilon :

Epsilon est une famille de langages de script basés sur Java pour automatiser les tâches courantes d'ingénierie logicielle basées sur des modèles, telles que la génération de code, la transformation de modèle à modèle et la validation de modèle, qui fonctionnent immédiatement avec EMF (y compris Xtext et Sirius), UML, Simulink, XML et autres types de modèles. Epsilon comprend également des éditeurs et des débogueurs basés sur Eclipse, des outils de réflexion pratiques pour la modélisation textuelle et la visualisation de modèles, ainsi que des tâches Apache Ant.²

1. <https://www.eclipse.org/Xtend/>

2. <https://www.eclipse.org/epsilon/>



FIGURE 2.9 – Les outils de transformation des modèles.

2.10.5.1 Les framework de transformation des modèles.

Dans cette section, nous allons présenter Les framework de transformation des modèles .

2.10.5.1.1 Ecore Eclipse :

Définition 11

Pour pouvoir construire des modèles avec un langage de modélisation, des outils sont nécessaires pour éditer et interpréter les modèles. Les ateliers des langages (Fowler, 2005) tel que l’« Eclipse modelling Framework » (EMF), « Metacase » (MetaEdit+), « Generic Modelling Environment» (GME) ou « outils DSL de Microsoft » réduisent de manière significative les efforts pour la construction des outils supports des langages de modélisation autour des métamodèles. Dans ce memoie nous utilisons EMF, puisqu’il représente la plateforme de modélisation la plus utilisée pour l’implémentation de EMOF. Essentiellement le langage Ecore définit un environnement de modélisation EMF basé sur EMOF. Son but est de permettre une génération de code automatique des interfaces et des classes d’un métamodèle pour pouvoir mettre à la disposition des éditeurs réflexifs (permettant les manipulations des concepts du métamodèle). la figure Figure 2.9 illustre quelques outils de transformation de modèles.

Définition 12

Le langage Ecore se distingue du langage MOF et EMOF en particulier par deux notions. Les associations ne sont pas représentées mais remplacées par des références directes entre les concepts. Pour générer du code, Ecore intègre également les notions de Java nécessaires par le concept d’EAnnotation. Les métaclasses Ecore sont toutes préfixées par la lettre ‘E’ (EClass, EAttribute, EString, etc.). La figure 1.6 présente un extrait du métamodèle Ecore utilisé lors des définitions de nouveaux métamodèles. Les concepts d’un nouveau langage de modélisation sont des EClass qui peuvent définir des propriétés (EAttribute), des opérations (EOperation) ou des relations (EReference) vers d’autres EClass. Un des principaux avantages de Ecore est sa simplicité et le grand nombre d’outils disponible. Actuellement Ecore est devenu le standard dans les plateformes de l’ingénierie dirigée par les modèles.

2.10.5.1.2 Xtext :

Xtext [[haase2007introduction](#)] permet le développement de la grammaire des langages spécifiques aux domaines (DSL : Domain Specific Languages) et d’autres langages textuels en utilisant une grammaire qui ressemble au langage EBNF (Extended Backus-Naur Form). Il est étroitement lié à l’Eclipse Modeling Framework (EMF). Il permet de générer un métamodèle Ecore, un analyseur syntaxique (parser, en

anglais) basé sur le générateur ANTLR³ ou JavaCC⁴ et un éditeur de texte sous la plate-forme Eclipse afin de fournir un environnement de développement intégré IDE spécifique au langage. La figure ?? fournit une vue d'ensemble de l'outil Xtext. La forme textuelle du DSL constitue le point de départ. La grammaire du DSL est le point d'entrée de l'outil. Xtext produit, en sortie, un analyseur syntaxique, un éditeur et un méta-modèle pour le DSL.

Xtext permet de considérer la forme textuelle du DSL comme un modèle conforme à la grammaire d'entrée, donc au méta-modèle généré. La figureFigure 2.10 illustre une vue d'ensemble de l'outil Xtext



FIGURE 2.10 – Vue d'ensemble de l'outil Xtext.

2.10.6 Quelques approches de transformation de modèle en détail

Définition 13

Les langages de transformation de modèles peuvent être classés en deux grandes catégories, selon la nature de la production de données : les transformations modèle à modèle (M2M) et modèle à texte (M2T). Dans ce contexte, divers langages de transformation de modèles sont inclus selon les deux catégories susmentionnées, tels que ATL (M2M) et Acceleo (M2T). Dans cette section, nous allons présenter la classification de la transformation des modèles proposée dans l'état de l'art (voir Figure Figure 2.11).

Définition 14

raducteur

Actuellement, il existe plus de 30 approches de transformation de modèles dans la littérature. Parmi ces approches, Czarnecki et Helsen distinguent :

2.10.6.0.1 Visitor-based :

Basé sur les visiteurs : un modèle de visiteur implémenté dans un langage de programmation traverse le modèle dans un framework orienté objet (par exemple, pour joli-imprimer une syntaxe concrète). Cela devient programme-ming plutôt que modelage.

2.10.6.0.2 Template-based :

Les modèles sont généralement exprimés dans la syntaxe concrète du modèle cible, pour donner des exemples avec des annotations de méta-code pour accéder au modèle source. Cette approche est souvent utilisée par des générateurs de code (par exemple, Enterprise Architect)

3. ANTLR (ANother Tool for Language Recognition) is a powerful parser generator for reading, processing, executing, or translating structured text or binary files.

4. Compilateur java-compilateur. Il s'agit d'un outil de générateur d'analyseur et d'analyseur lexical populaire open source développé par Oracle Corporation.

2.10.6.0.3 Direct-Manipulation :

les modèles proposent une API⁵ pour les exploiter.(décrit dans le méta-méta-langage) pour manipuler des modèles.Cela reste de la programmation tout en étant conscient qu'il s'agit de modèles avec une API dédiée.

2.10.6.0.4 Operational :

Ils consistent en des langages modélisés qui permettent de manipuler des modèles par exemple, déclaratifs et/ou impératif OCL⁶ De plus, les méta-modèles sont complétés par une construction impérative offrant des méthodes/fonctions appellés dans les modèles eux-mêmes.

2.10.6.0.5 Graph transformation-based :

les modèles sont représentés sous forme de graphiques, ainsi la théorie de la transformation de graphe est utilisée pour transformer les modèles.c'est une manière déclarative de décrire les opérations sur les modèles.

2.10.6.0.6 Relational :

Ils décrivent discrètement les correspondances entre la source et la cible.Souvent sous la forme de contraintes à résoudre.Ils sont implicitement multidirectionnels, mais la transformation sur place est plus difficile à réaliser.

2.10.6.0.7 Hybrid :

C'est une combinaison de deux ou plusieurs des approches précédentes.

2.10.7 La classifications des modèles de transformation

Ce qui différencie les différentes approches permettant l'élaboration des transformations de modèles est la façon dont sont spécifiées les règles des transformations. Six approches aujourd'hui répertoriées permettent la spécification de ces règles de correspondances.

- **Apprcohe par programmation :** Consiste à utiliser les langages de programmation orientée objet. L'idée est de programmer une transformation de modèles de la même manière que l'on programme n'importe quelle application informatique. Ces transformations sont donc des applications informatiques qui ont la particularité de manipuler des modèles. Ces applications informatiques utilisent les interfaces de manipulation de modèles. Cette approche est la plus utilisée car elle est très puissante et fortement outillée.

5. Une API, ou interface de programmation d'application, est un ensemble de définitions et de protocoles qui facilite la création et l'intégration de logiciels d'applications.

6. OCL : (Object Constraint Language) est un langage informatique d'expression des contraintes utilisé par UML.

- **Apprcohe Impérative :** On décrit ce que l'on fait mécaniquement, il n'y a aucune part de choix qui n'appartient pas au développeur. Par exemple une requête SQL n'est pas impérative, le SGBD fait comme il veut pour récupérer les données, de même pour le HTML, le navigateur internet est libre d'interpréter la page comme il veut .
- **approches hybride :** Les approches hybrides sont une combinaison des différentes techniques. On peut notamment retrouver des approches utilisant à la fois des règles à logique déclarative et des règles à logique impérative.
- **Apprcohe par API :** Certaines APIs (Application Programming Interface) de transformation sont décrites dans des langages de programmations impératifs et sont ensuite fournies comme des bibliothèques permettant la description du processus de transformation suivant la syntaxe du langage utilisé. Les résultats sont alors d'une performance considérable. Cependant, le développeur a la charge de l'organisation et de la description de toutes les étapes de manière explicite en termes de déclarations impératives [[cuong2011model](#)]
- **Apprcohe par Modélisation :**
Consiste à appliquer les concepts de l'ingénierie des modèles aux transformations des modèles elles-mêmes. L'objectif est de modéliser les transformations de modèles et de rendre les modèles de transformation pérennes et productifs et d'exprimer leur indépendance vis-à-vis des plates-formes d'exécution .Cette approche est actuellement au stade de la recherche. Le standard MOF2.0 QVT⁷ en cours de définition à l'OMG a pour objectif de définir le métamodèle permettant l'élaboration des modèles de transformation de modèles. Actuellement, seuls quelques prototypes de recherche supportent cette approche.
- **Apprcohe par graphe :** Cette catégorie d'approches de transformation de modèles s'appuie sur la théorie de graphes. Une approche par transformation de graphes utilise usuellement des graphes typés, orientés et étiquetés où les règles de transformation de graphes possèdent un schéma LHS et un schéma RHS : le schéma LHS est repéré dans le modèle source et remplacé par le schéma RHS dans le modèle cible.
- **Apprcohe Déclarative :** [[dehayni2009some](#)] Dans cette approche, Une règle fait la correspondance entre un ensemble de concept invoqué au niveau du modèle source et un ensemble de concept qui devrait être adopté au niveau du modèle cible. La mise en œuvre est réalisée par un moteur d'inférence. Cependant, l'un des principaux inconvénients de cette approche est la charge élevée de travail que doit effectuer le développeur qui doit spécifier toutes les contraintes d'appui à la transformation, tâche qui s'avère souvent fastidieuse
- **Apprcohe par template :** Consiste à définir les canevas des modèles cibles souhaités en y déclarant des paramètres. Ces paramètres seront substitués par les informations contenues dans les modèles sources. On appelle ces canevas des « modèles cibles paramétrés » ou des « modèles Template ». L'exécution d'une transformation consiste à prendre un modèle Template et à remplacer ses paramètres par les valeurs d'un modèle source. Cette approche nécessite un langage particulier permettant la définition des modèles Template. Sa puissance réside principalement dans le langage de définition des modèles Template. De tels langages sont en cours d'élaboration et n'ont pas encore la maturité des langages de programmation orientée objet utilisés dans la première approche . Le tableau suivant [Tableau 2.1](#) montre les approches de langages de la transformation des modèles.

7. Query, View, Transformation

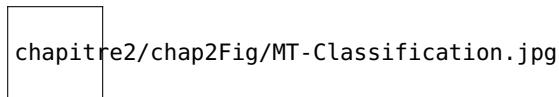


FIGURE 2.11 – Approches de transformation de modèle.

	Approches utilisée	Transformation (M2M)	Transformation(M2T)
AndroMDA	Template	+	+
ATL	Hybride	+	-
Accéléo	Template	-	+
Mola	Impératif	+	-

TABLEAU 2.1 – Approches de langages de la transformation des modèles.

2.10.7.1 Topologie de transformation

Une transformation de modèle génère un modèle cible à partir d'un modèle source. ces deux modèles source et cible sont conformément aux même métamodèle ou à deux métamodèles différents aussi les modèles sources et cibles peuvent appartenir au même niveau d'abstraction (raffinement) ou à deux niveaux d'abstractions différents

2.10.7.1.1 Transformation endogène

Une transformation de modèles est dite endogène c'est lorsque les modèles source et cible sont conforme au même métamodèle. Donc les règles de transformation sont présentées sur un seul métamodèle. On utilise les transformations endogènes dans le cadre d'optimisation de modèle ou d'une simplification d'un modèle ou dans un refactoring.

2.10.7.1.2 Transformation exogène

Une transformation de modèles est dite exogène c'est lorsque le métamodèle du modèle source est différent du métamodèle du modèle cible. Donc le formalisme qui va exprimer le modèle source est différent du formalisme qui va exprimer le modèle cible. On utilise les transformations exogènes dans le cadre de migration de modèle d'une plateforme à une autre en restant au même niveau d'abstraction, ou pour la génération de code et aussi les opérations de rétro-ingénierie ou rétro-conception.

2.10.7.1.3 Verticalité et horizontalité

Une transformation de modèle peut aussi être classer selon un autre critère qui est le niveau d'abstraction si les modèles cible et source appartiennent au même niveau d'abstraction donc on a une transformation horizontale et si les modèles cible et source appartiennent à deux niveaux d'abstractions différents donc on a une transformation verticale. Par exemple, " une génération de code est une transformation verticale"

2.10.7.2 Exactitude de la transformation du modèle

L'extraction de modèle est une transformation de modèle exogène et verticale. L'extraction est l'inverse de la génération de code et c'est l'une des activités essentielles dans l'ingénierie inverse et de la compréhension du programme . Elle permet de construire un modèle visuel à un niveau d'abstraction supérieur qui permettra de connaître la structure du code .

2.11 Conclusion

La transformation des modèles est l'élément cœur de l'ingénierie dérivée par les modèles. Dans ce chapitre, nous avons présenté les généralités sur la transformations des modèles. Ces notions de base nous permettent de se familiariser avec ce domaine. La transformation des modèles est souvent exprimée avec des règles déclaratives qui contiennent des préconditions, des actions et des postconditions. Dans certaines situations la connaissance du domaine est immature, par conséquent il n'existe pas une règle prédefinie. La transformation des modèles doit reposée sur une séries d'expériences, ce qu'on appelle la transformation des modèles à base d'exemple (MTBE) qui sera l'objet du chapitre suivant.



Les approches de Transformation des modèles par l'exemple

« You cannot control what you cannot measure. »
— Tom DeMarco (Software Engineer) (1863-1947)

Sommaire

3.1	Introduction	26
3.2	Pourquoi L'exemple	26
3.3	Formalisation de modèle de transformation par l'exemple	27
3.4	Classification des approches MTBE	27
3.4.1	Approche explicite	27
3.4.2	Approche implicite	27
3.5	Bilan des travaux antérieurs sur les travaux MTBE	27
3.5.1	Le problème de déduction de la règles de la transformation	27
3.5.2	Incertitudes liées à la maturité de domaine de connaissance	27
3.5.3	Méthodes de résolution	27
3.5.4	Synthèse	27
3.5.5	Conclusion	27
3.6	Concepts de base de TMPE	28
3.6.1	Motivation	28
3.6.2	Pourquoi l'exemple ?	29
3.6.3	Derivation de la règle de transformation	29
3.6.4	Les approches de Transformation de modèle a base d'exemples	32
3.6.5	Carte conceptuelle des approches MTBE (Approches Timeline)	35
3.6.6	Synthèse	37
3.7	Conclusion	37

3.1 Introduction

3.2 Pourquoi L'exemple

2.4.1 Algorithms and Applications

2.3 Why We Benefit from Examples?	13	2.4 Common By-Example
Approaches	14	2.4.1 Query By Example
15	2.4.2 Programming By-Example	17
By-Example	18	2.4.3 Web-Scheme Transformers
2 Background	8	2.5 Summary and Further Reading
2.3 Model Transformation By-Example	2.4 Machine Learning	
Techniques	2.4.1 Algorithms and Applications	21
Inductive Logic Programming	23	2.4.2
.....	26

3.3 Formalisation de modèle de transformation par l'exemple

3.4 Classification des approches MTBE

3.4.1 Approche explicite

3.4.1.1 Travaux de xxxxx et a

3.4.1.2 Travaux de xxxxx et a

3.4.1.3 Travaux de xxxxx et a

3.4.1.4 Travaux de xxxxx et a

3.4.1.5 Travaux de xxxxx et a

3.4.1.6 Travaux de xxxxx et a

3.4.1.7 Travaux de xxxxx et a

3.4.1.8 Travaux de xxxxx et a

3.4.1.9 Travaux de xxxxx et a

3.4.2 Approche implicite

3.4.2.1 Travaux de xxxxx et a

3.4.2.2 Travaux de xxxxx et a

3.4.2.3 Travaux de xxxxx et a

3.4.2.4 Travaux de xxxxx et a

3.4.2.5 Travaux de xxxxx et a

3.4.2.6 Travaux de xxxxx et a

3.4.2.7 Travaux de xxxxx et a

3.4.2.8 Travaux de xxxxx et a

3.5 Bilan des travaux antérieurs sur les travaux MTBE

3.5.1 Le problème de déduction de la règles de la transformation

3.5.1.1 Problèmes adjacents à la déduction de ka règles

27

3.5.2 Incertitudes liées à la mmaturité de domaine de connaissance

3.5.3 Méthodes de résolution

3.5.3.1 Méthodes basées recherche /optimisation

.....37 2.3.1.1. Problèmes adjacents au HHCRSP.....	37
2.3.2. Incertitudes liées au HHCRSP opérationnel.....	38
2.3.3. Méthodes de résolution.....	39
2.3.3.1. Méthodes exactes.....	39
2.3.3.2. Méthodes approchées	41
2.3.3.3. Approches hybrides.....	44
2.3.3.4. Approches liées aux incertitudes opérationnelles.....	44
2.3.4. Synthèse.....	46
3. CONCLUSIONS.....	47

L'objectif de ce chapitre est d'explorer les travaux qui traitent l'apprentissage de transformations dans le cadre de la transformation des modèles par l'exemple (TMPE, en anglais, MTBE : Model Transformation By Example). Ce chapitre est consacré à l'apprentissage des transformations de modèles. Nous allons passer en revue les différentes contributions qui s'inscrivent dans le cadre de TMPE (*revue de littérature non exhaustive*). Nous verrons deux familles d'approches d'apprentissage existantes dans la littérature, à savoir, les approches d'apprentissage par l'exemple et celles par démonstration. Enfin, la dernière partie de ce chapitre prend la forme d'une synthèse qui résume les caractéristiques clés et les limites des contributions identifiées.

3.6 Concepts de base de TMPE

Dans cette section, nous allons présenter les concepts de base de la transformation des modèles à base d'exemples.

3.6.1 Motivation

L'écriture de transformations de modèles est une tâche difficile qui nécessite parfois beaucoup d'efforts et de temps. Dans ce sens, la transformation de modèle par l'exemple (TMPE) semble être une solution appropriée à cette problématique. De manière analogue à la programmation par l'exemple[21] et à l'interrogation par l'exemple[42] , les approches de TMPE ont pour but d'apprendre automatiquement un programme de transformation à partir d'artéfacts fournis en guise d'exemples. Il existe deux axes de recherche portant sur l'apprentissage de transformations de modèles par l'exemple : les approches par l'exemple proprement dites (TMPE), ainsi que celles par démonstration (TMPD).

Nous explorons dans ce chapitre la première catégorie, tandis que nous aborderons la seconde dans la section suivante. Les travaux de TMPE oeuvrent à dériver automatiquement un programme de transformation à partir d'un ensemble d'exemples fournis en entrée. Chaque exemple est une paire de modèles constituée d'un modèle source et d'un modèle cible. En plus des exemples, la majorité des approches de TMPE exploitent des traces fines de transformation. Ces traces sont des liens, de plusieurs-à-plusieurs, qui associent un groupe de n éléments sources à un groupe de m éléments cibles.

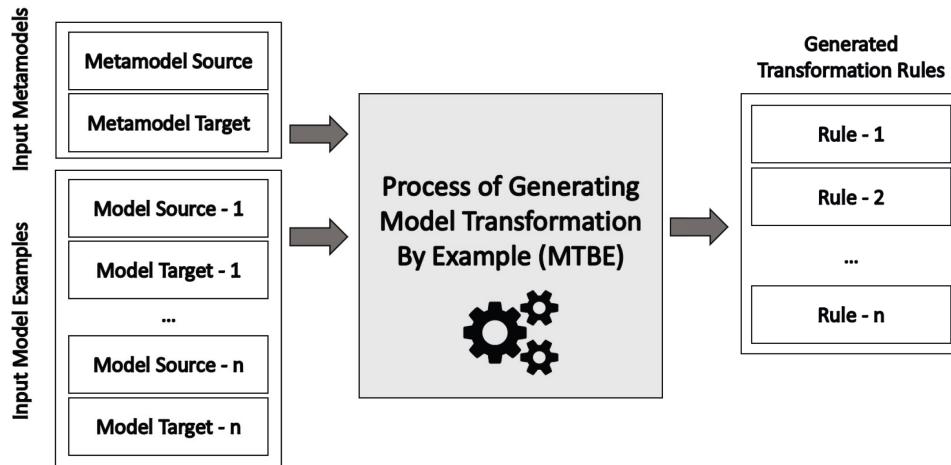


FIGURE 3.1 – Principe de transformation de modèle par l'exemple [20].

3.6.2 Pourquoi l'exemple ?

Les exemples jouent un rôle clé dans le processus d'apprentissage humain. Il existe de nombreuses théories sur les styles d'apprentissage dont certaines exemples doivent être considérés comme des artefacts majeurs.

Pour une description des théories de style d'apprentissage populaires d'aujourd'hui, voir par exemple. Toutes ces théories, en particulier celles référencées, sont assez similaires en ce qui concerne leur notion d'une dimension d'apprentissage utile à la déduction de règles générales, comme illustré dans la Figure Figure 3.2. Aussi, les exemples sont des outils courants dans l'enseignement de l'informatique. Surtout dans les domaines fortement influencés par les algorithmes et les mathématiques.

La Figure Figure 3.2 montre une sorte de transformation basée sur une série des exemples. Dans cet exemple montre un exemple de paire avec trois traces identifiées (toutes les traces ne sont pas affichées). Pour chaque trace, le fragment cible (en bas) a été déterminé comme correspondant au fragment source (en haut). Les cartographies de transformation peuvent être identifiées par un expert au cours du processus de conception ou récupérées (semi-) automatiquement à l'aide d'un traitement informatique basé sur l'exemple.

3.6.3 Déivation de la règle de transformation

Dans cette section, nous allons présenter une classification des approches transformations adoptées par la communications de l>IDM. Nous distinguons deux grandes familles (i) Approches basées sur les règles et (ii) Approches basées sur les exemples.

- Approches basées sur les règles :** Les Transformations à base de règles sont souvent décrites comme des règles en premier lieu. Ce type de transformation est un langage de programmation déclaratif à base de règles basé sur le mode impératif ou déclaratif. Voici un exemple de transformation inplace (UML 2 UML). La Figure Figure 3.3 représente une classe UML nommée Person qui doit être transformée en classe nommée Contact. Le modèle d'entrée qui contient la classe

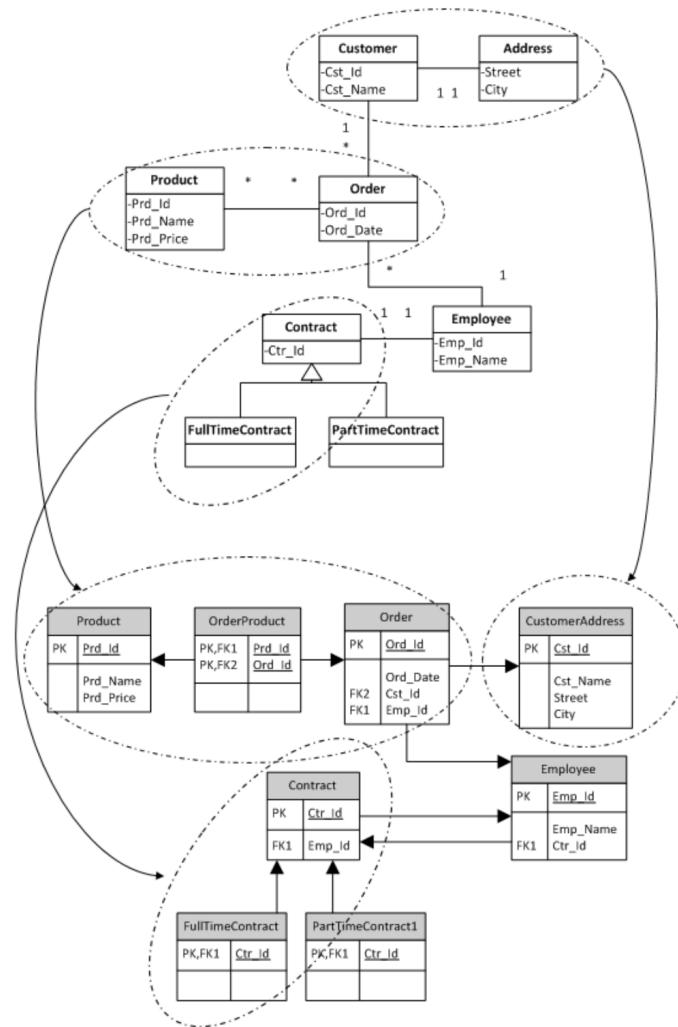


FIGURE 3.2 – Exemple d'un exemple de paire source-cible avec trois traces identifiées

Person est stocker dans un fichier XMI (`Ma.xmi`). La transformation exprimé en langage ATL genere un fichier de sortié nommé `Mb.xmi` qui contient la description de classe Contact.

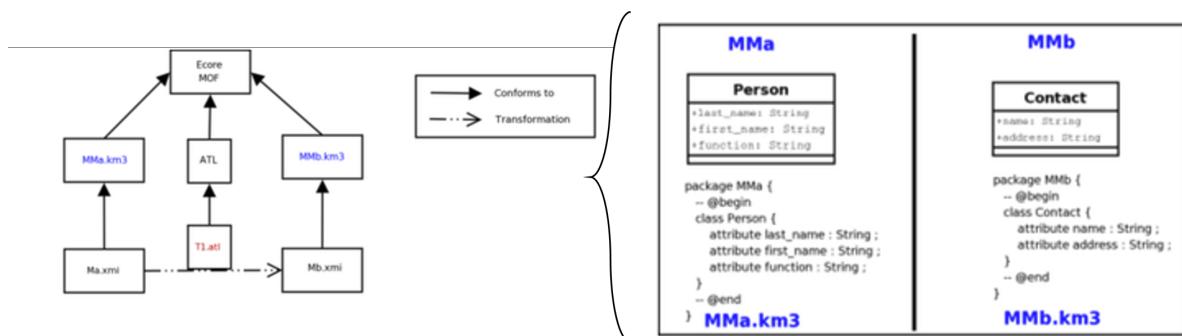


FIGURE 3.3 – Illustration de la transformation

Listing [Liste 3.1](#) montre le programme de la transformation qui est stocké aussi dans un fichier

ATL T1.alt.

```

1 T1.alt
2 -----
3 module Person2Contact
4 create OUT : MMb from IN : MMa;
5
6 rule Start {
7 from
8 p : MMa!Person (
9 p.function = 'Boss'
10 )
11 to
12 c : MMb!Contact (
13 name ← p.first_name + p.last_name
14 )
15 }

```

Listing 3.1 – Exemple d'une règle de transformation inplace (UML2UML) exprimée en ATL

2. **Approches basées sur les exemples :** Cette catégorie regroupe deux types de méthodes qui se trouvent dans la littérature : (i) les approches basées sur des algorithmes de recherche (par ex. *algorithme génétique*) et les algorithmes de l'apprentissage automatique (par ex. *arbre de décision*).
– **Approches basées sur la recherche :** ces approches considèrent le problème de transformation des modèles comme un problème d'optimisation sous contraintes. La stratégie de recherche utilisée est basée sur une fonction objective qui doivent être définie par le concepteur de modèle de transformation, prenant en considération des contraintes de l'application et les besoins des utilisateurs. Nous pouvons citer deux algorithmes utilisés : l'algorithme génétique et les algorithmes d'optimisation à base de PSO (Particle Swarm Optimization).
– **Approches basées sur l'apprentissage :** dans cette catégorie on distingue deux types de techniques : implicite et explicite. Les approches explicites permettent de générer les règles de transformation avec une description détaillée qui conduit à la génération de modèle cible. Les algorithmes les plus utilisés sont l'arbre de décision, la programmation en nombre entier etc.. Pour les techniques implicites, nous pouvons citer les algorithmes de deep learning ou apprentissage profond, Natural language processing (NLP).

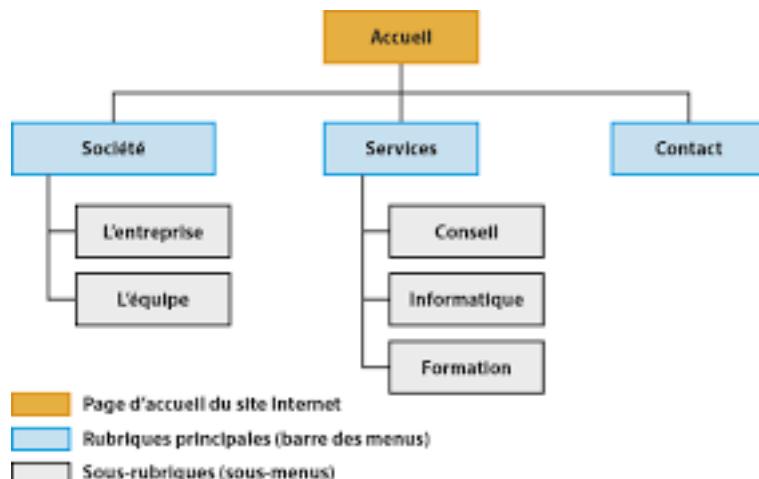


FIGURE 3.4 – Classification des approches de transformation des modèles.

Dans la section suivante, nous allons présenter quelques travaux de la littérature qui sont basés sur le MTBE.

3.6.4 Les approches de Transformation de modèle à base d'exemples

Dans explorons l'état de l'art, nous avons recencé les travaux qui traitent la transformation desmodèle à base l'exemple.

3.6.4.1 Travaux de Varro

En 2006, Varró [40] propose une première approche pour l'apprentissage de transformations à partir de paires d'exemples en utilisant un algorithme ad-hoc basé sur les graphes. L'algorithme prend en entrée des paires de modèles interreliés (accompagnés de traces) et dérive un ensemble de règles de transformation de type 1 1. Le processus de dérivation est itératif et interactif, à chaque itération, les règles produites sont ranées par l'utilisateur. La contribution est ensuite automatisée davantage par Balogh et al.[3] en utilisant la programmation logique inductive (PLI). Cette nouvelle approche, bien que toujours itérative et incrémentale, permet de dériver des règles de transformation plus complexes (de type n, m).

3.6.4.2 Travaux de Wimmer

Durant la même période Wimmer et al. [[gierlinger2003rapide](#), [al1995maternelle](#)] propose une approche similaire pour dériver des transformations sous la forme de règles de type 1 1 exprimées en ATL. La contribution en question est également itérative et incrémentale, elle dière cependant des deux contributions précédentes sur deux aspects. D'abord, les traces de transformation exploitées sont spéciées dans une syntaxe concrète plutôt qu'abstraite. Ensuite, le processus de dérivation se fait selon une approche orientée objet, contrairement à celui de Varró où des graphes sont utilisés. La contribution de Wimmer est également améliorée dans des contributions subséquentes, par Strommer et al. [[bodmer1989immunosuppression](#)], où un langage pour la déinition de correspondances de type (n , m) est présenté, jumelé à un algorithme de raisonnement pour la dérivation de règles (nm également) à partir des correspondances exprimées. L'usage d'opérateurs de chaînes tel que la concaténation est également brièvement mentionné dans l'une des deux contributions.

La figure ?? montre un méta-modèle simplifiés de UML et Entité-Relation utilisé dans Wimmer .

3.6.4.3 Travaux de Garcia-Magarino

Un autre travail qui s'inscrit dans le contexte des transformations de modèles par l'exemple est celui de Garcia-Magarino et al. [[khan2021deep](#)] qui propose un algorithme permettant de générer des règles de transformation de type (n ,m), vraisemblablement dans plusieurs langages de transformation, à partir d'un ensemble de paires de modèles sources et cibles interconnectés (agrémentés par des traces). À l'instar de [40], l'approche utilise des graphes. Afin de pallier le fait que certains langages de transformations ne permettent pas d'écrire des règles de plusieurs à plusieurs, les règles sont d'abord

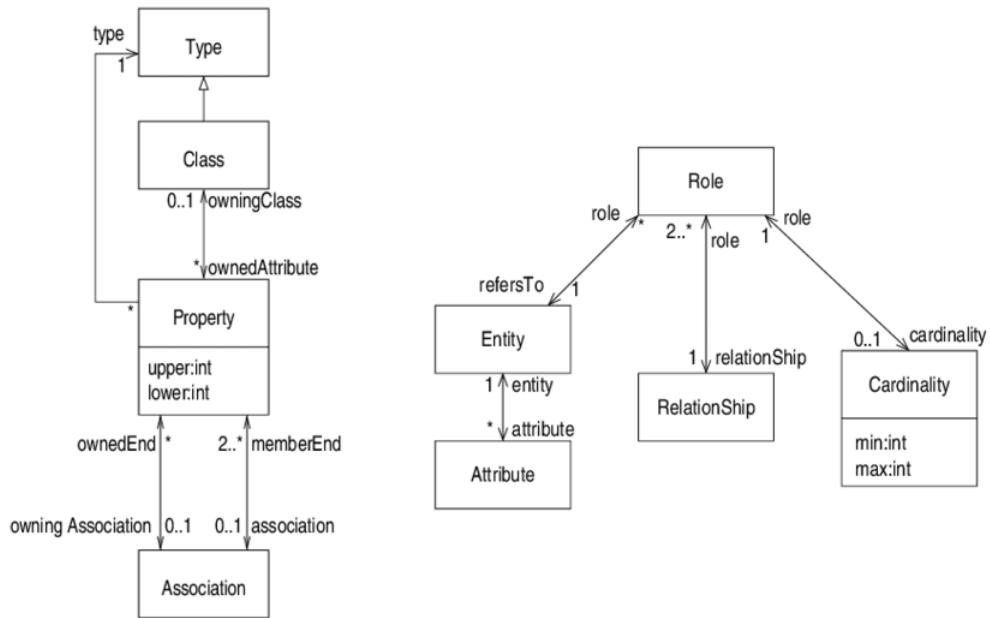


FIGURE 3.5 – Méta-modèles simplifiés de UML (à gauche) et Entité-Relation (à droite) utilisés dans Wimmer et al 2007 [15].

dérivées dans un langage de transformation générique avant d'être transformées dans le langage de transformation souhaité (ATL dans l'article).

3.6.4.4 Travaux de Kessentini

Kessentini et al. [40] utilise des analogies pour effectuer une transformation. Contrairement aux contributions citées plus haut, cette approche ne produit pas de règles de transformation, mais dérive plutôt le modèle cible directement à partir du modèle source en considérant la transformation de modèles comme un problème d'optimisation. Le problème tel que posé est abordé en utilisant l'optimisation par essaims particulaires (OEP) dans la première contribution et une combinaison de OEP et du recuit simulé (RS) dans la deuxième. L'approche d'apprentissage est ensuite améliorée dans [kessentini2010generating] où des règles de transformation sont produites à partir de modèles sources et cibles qui ne sont pas accompagnés de traces de transformation. L'approche est validée sur une transformation de modèles comportementaux (diagramme de séquence vers réseau de Petri colorés).

3.6.4.5 Travaux de Dolques

Une autre contribution de TMPE qui ne produisait pas de règles de transformation initialement est celle de Dolques et al. [16]. Ce travail se base sur l'analyse relationnelle de concepts (ARC), une variante de l'analyse formelle de concepts, pour classer les éléments sources et cibles ainsi que les correspondances fournies en entrée. Les patrons identifiés sont organisés dans des treillis partiellement ordonnés et sont ensuite analysés pour filtrer les plus pertinents. L'approche est également étendue par Saada et al. [34] où des règles exécutables sont produites à partir des patrons filtrés. Dans cette dernière

contribution es règles sont exprimées en Jess.⁸ Il est possible ensuite de raisonner sur ces connaissances à l'aide de règles déclaratives. Jess applique les règles sur la base de faits en utilisant un ltrage par motifs basé sur l'algorithme Rete [**forgy1989rete**]. Il est possible d'associer la notion de fait en Jess au concept d'objet dans le paradigme orienté objet (POO). Chaque fait peut contenir plusieurs attributs, appelés « slots ». Jess ore également la possibilité de dénier des gabarits de faits (template) qui correspondent à la notion de classe dans le POO. An d'utiliser Jess pour la transformation de modèles,nous exprimons chaque modèle sous la forme d'un ensemble de faits. Chaque élément du modèle est un fait dont les slots sont des attributs ou des références vers d'autres éléments. Les méta-modèles sont exprimés, quant à eux, sous la forme de gabarits de faits. La Liste 3.1 présente un exemple trivial, où sont représentés, un méta-modèle et un modèle comportant un seul élément.

3.6.4.6 Travaux de Baki

[2] cet approche prend en entrée un ensemble de pairs d'exemples de modèles sources et cibles accompagnés de traces de transformations capturant la correspondance entre les différents fragments sources et cibles. Le processus d'apprentissage proposé dans cette dernière permet alors de dériver un programme de transformation en trois étapes principales. Dans un premier temps, les traces fournies sont complétées et analysées afin de construire des pools d'exemples cohérents. Ensuite, un ensemble de règles est dérivé pour chaque pool en utilisant un programme génétique. Durant cette étape, les règles sont également traitées pour éliminer d'éventuelles erreurs ou incohérences. Enfin, la troisième et dernière étape consiste à fusionner les groupes de règles apprises en un programme de transformation qui est affiné en utilisant la méthode du recuit simulé. Le processus d'apprentissage de cette approche est validé sur sept (7) cas de transformations pré- sentant divers caractéristiques et degrés de complexité. Les résultats obtenus indiquent que les transformations les plus communes sont parfaitement apprises. Par ailleurs, l'apprentissage des cas de transformations les plus complexes permet de générer des modèles cibles très similaires à ceux attendus.

3.6.4.7 Travaux de Faunes

Les travaux les plus récents dans le contexte de la TMPE sont ceux de Faunes et al.[18, 19] dans les quels la programmation génétique (PG) est utilisée pour faire évoluer une population de transformations sur plusieurs générations jusqu'à produire la transformation attendue. Le processus de dérivation prend en entrée des paires d'exemples de modèles sources et cibles uniquement (sans traces de transformation) et produit en sortie des règles exécutables de type (n, m). L'approche est ensuite améliorée par la contribution de Baki et al.[2] où le programme génétique tente d'apprendre simultanément les règles de transformation ainsi que le contrôle d'exécution qui doit être exercé sur cellesci pour former un programme de transformation correct. Cette seconde version permet également de dériver des règles de transformations plus complexes en incluant la négation de conditions, l'usage de primitives de navigation ainsi que la considération des types et des domaines de déinition lors de la construction du modèle cible.

Le tableau suivant **Tableau 3.1** illustre les caractéristiques des approches MTPE actuelles.

8. Jess :C'est un environnement de transformation de modèles.Jess est un moteur de règles qui permet de stocker en mémoire des connaissances exprimées sous la forme de faits (facts).

Approche	Algorithm	Entrée	Sortie
Varró	Ad-hoc	Exemples et Traces	Règles
Wimmer	Ad-hoc	Exemples et Traces	Règles
Balogh	PLI	Exemples et Traces	Règles
Kassentini	OEP/OEP-RS	Exemples	MC
Deloques	ARC	Exemples et Traces	Règles
Faunes	PG	Exemples	Règles
Baki	PG	Exemples	Règles

TABLEAU 3.1 – Caractéristiques des approches TMPE actuelles.

3.6.5 Carte conceptuelle des approches MTBE (Approches Timeline)

Les travaux de TMPE ont dérivé automatiquement un programme de transformation à partir d'un ensemble d'exemples fournis en entrée. Chaque exemple est une paire de modèles constituée d'un modèle cible (MS) et d'un modèle source (MC). En plus des exemples, la majorité des approches de TMPE exploitent des traces de transformation. Ces traces sont des liens, de plusieurs-à-plusieurs, qui associent un groupe de n éléments sources à un groupe de m éléments cibles.

Nous avons essayé de tracer l'évolution de ce domaine sous forme d'une carte pour montrer les travaux de bases qui constituent l'état de l'art. La Figure ?? montre la carte conceptuelle des approches MTBE (Approches Timeline).

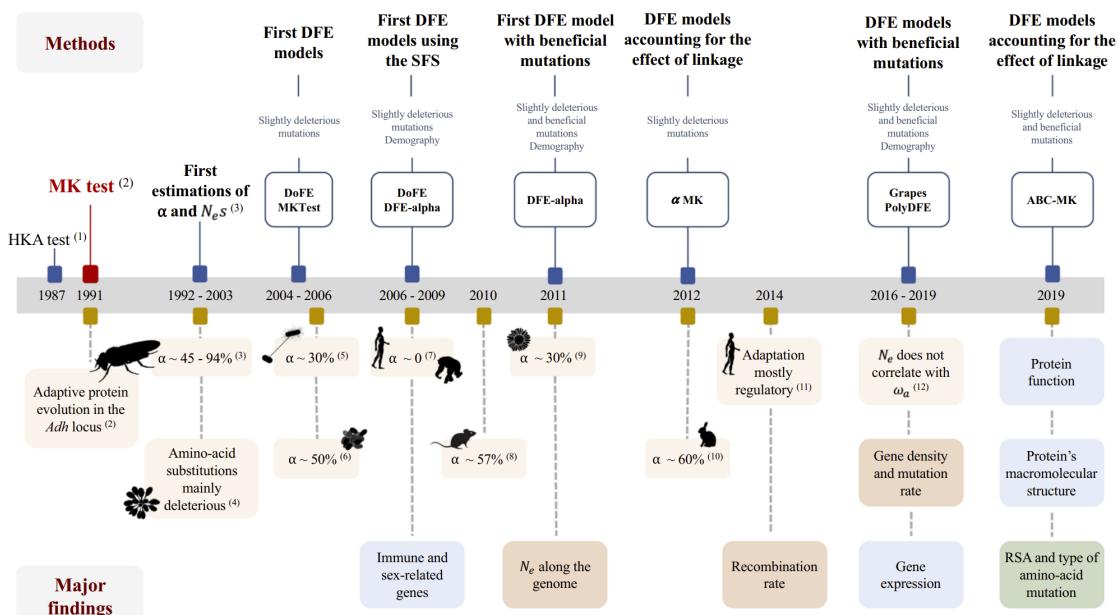


FIGURE 3.6 – Carte conceptuelle des approches MTBE (Approches Timeline)

En 2006, Varró propose une première approche pour l'apprentissage de transformations à partir de paires d'exemples en utilisant un algorithme basé sur les graphes. L'algorithme prend en entrée des paires de modèles inter reliés (accompagnés de traces) et dérive un ensemble de règles de transformation

de type 1-1. Le processus de dérivation est itératif et interactif, à chaque itération, les règles produites sont ranées par l'utilisateur.

La contribution est ensuite automatisée davantage par Balogh et al en utilisant la Programmation Logique Inductive (PLI). Cette nouvelle approche est itérative et incrémentale, permet de dériver des règles de transformation plus complexes (de type $n\text{-}m$).

Durant la même période Wimmer propose une approche similaire pour dériver des transformations sous la forme de règles de type 1-1 exprimées en ATL (*ATL Transformation Language*). La contribution en question est également itérative et incrémentale, elle dépend des deux contributions précédentes sur deux aspects. D'abord, les traces de transformation exploitées sont spécifiées dans une syntaxe concrète plutôt qu'abstraite. Ensuite, le processus de dérivation se fait selon une approche orientée objet [1, 20, 31, 29].

La contribution de Wimmer est également améliorée dans des contributions subséquentes, par Strommer, où un langage pour la définition de correspondances de type $n\text{-}m$ est présenté, jumelé à un algorithme de raisonnement pour la dérivation de règles (nm également) à partir des correspondances exprimées. Un autre travail qui s'inscrit dans le contexte des transformations de modèles par l'exemple est celui de Garcia-Magarino qui propose un algorithme permettant de générer des règles de transformation de type $n\text{-}m$, vraisemblablement dans plusieurs langages de transformation, à partir d'un ensemble de paires de modèles sources et cibles interconnectés (agrémentés par des traces).

Kessentini utilise des analogies pour effectuer une transformation. Contrairement aux contributions citées plus haut, cette approche ne produit pas de règles de transformation, mais dérive plutôt le modèle cible directement à partir du modèle source en considérant la transformation de modèles comme un problème d'optimisation. Le problème tel que posé est abordé en utilisant l'optimisation par essaims particulaires (OEP ou PSO en anglais) dans la première contribution et une combinaison de OEP et du recuit simulé (RS) dans la deuxième.

L'approche d'apprentissage est ensuite améliorée où des règles de transformation sont produites à partir de modèles sources et cibles qui ne sont pas accompagnés de traces de transformation. L'approche est validée sur une transformation de modèles comportementaux (diagramme de séquence vers réseau de Petri colorés). Une autre contribution de TMPE qui ne produisait pas de règles de transformation initialement est celle de Dolques.

Ce travail se base sur l'analyse relationnelle de concepts (ARC), une variante de l'analyse formelle de concepts, pour classer les éléments sources et cibles ainsi que les correspondances fournies en entrée. Les patrons identifiés sont organisés dans des treillis partiellement ordonnés et sont ensuite analysés pour filtrer les plus pertinents.

L'approche est également étendue par Saada où des règles exécutables sont produites à partir des patrons ltrés. Dans cette dernière contribution, les règles sont exprimées en Jess[**faridmoayer2017optimisation**].

Les travaux les plus récents dans le contexte de la TMPE sont ceux de Faunes dans lesquels la programmation génétique (PG) est utilisée pour faire évoluer une population de transformations sur plusieurs générations jusqu'à produire la transformation attendue. Le processus de dérivation prend en entrée des paires d'exemples de modèles sources et cibles uniquement (sans traces de transformation) et produit en sortie des règles exécutables de type $n\text{-}m$.

L'approche est ensuite améliorée par la contribution de Baki où le programme génétique tente d'apprendre simultanément les règles de transformation ainsi que le contrôle d'exécution qui doit être exercé sur celles-ci pour former un programme de transformation correct. Cette seconde version permet également de dériver des règles de transformations plus complexes en incluant la négation de conditions, l'usage de primitives de navigation ainsi que la considération des types et des domaines de définition lors de la construction du modèle cible.

3.6.6 Synthèse

Nous avons synthétisé les travaux qui traitent le problème de transformation de modèles par l'exemples. Ces travaux sont basés sur des approches heuristiques comme par exemple les algorithmes génétiques [2, 19], des approches basées sur les graphes et des approches basées sur l'apprentissage automatique. Ces travaux traitent la problématique liée à la transformation de modèles comme l'absence des règles.

Les approches MTBE existantes ne résolvent que partiellement le problème de dérivation des règles de transformation. La plupart d'entre eux nécessitent des mappings détaillés (traces de transformations) entre les modèles source et cible des exemples, difficiles à fournir dans certaines situations. D'autres peuvent difficilement dériver des règles de transformation qui testent de nombreuses constructions dans le modèle source et/ou produisent de nombreuses constructions dans le modèle cible. Cependant, dans certains domaines comme *la robotique* et *l'automobile*, la connaissance elle non mature, cette nature de connaissance rend la transformation du modèle plus complexe et on peut trouver un multi points de vue. De plus, le manque d'ensemble de données au départ peut être un autre défi qui doit être relevé. Dans ce mémoire, nous abordons ce problème pour générer les règles de transformation en commençant sans jeu de données, avec les règles pouvant être affinées lors de l'utilisation du système en cours de conception ou de simulation.

Cette situation nous a motivé de voir la possibilité de proposer une approche de transformation inspirée de l'apprentissage par renforcement (*Reinforcement Learning*) qui raffine au fur et à mesure les règles de transformation.

3.7 Conclusion

Dans ce chapitre, nous allons présenter les approches de MTBE qui représente un bref état de l'art des différentes approches et les algorithmes/techniques utilisés pour permettre de déduire si un modèle source peut en transformer par un modèle cible dans le cas de l'absence de la règle. Le chapitre suivant présente notre approche de transformation de modèle dans la cadre MTBE.

Troisième partie

Contributions



Cadre fondé sur les agents BDI pour la transformation collaborative des modèles à base de l'exemple

« Chaque individu apporte au monde sa contribution unique. »

— Jack Kornfield

Sommaire

4.1	Introduction	42
4.2	MoTrans-BDI Framework and its Theoretical Foundations	42
4.2.1	The MoTrans-BDI System Overview	42
4.2.2	Theoretical Foundation of MoTrans-BDI Framework	43
4.2.3	Conceptual Organization of MoTrans-BDI Framework	44
4.2.4	The MoTrans-BDI Process	46
4.3	Conclusion	52

4.1 Introduction

4.2 MoTrans-BDI Framework and its Theoretical Foundations

We present in this section our approach. MoTrans-BDI is intended to bring a flexible MTBE based on EMAS and Contract Net Protocol architecture. The BDI agents' behavior is inspired from the cognitive theory [10], thus they have the ability to explore the solutions space in a smart way by using the traditional operators of the genetic algorithms. We first give the foundations behind the MoTrans-BDI framework. Then, we describe its process and the involved components.

4.2.1 The MoTrans-BDI System Overview

We adopt a new MTBE approach that produces TMs from different experts' design views. Figure 4.1 presents the way of transforming a given source Model SM_i into its appropriate target model TM_i even if the source and target metamodels are different (e.g., transforming UML class diagrams to ER diagrams). MoTrans-BDI proposes a novel approach based on the BDI agents reasoning (Beliefs, Desires, Intentions) in a context of Contract Net Protocol [32] allowing to build diverse conceptual views on any part of the source model that conforms to a given metamodel. It provides a two-step approach that explicitly separates the gathering of conceptual views from different transformer agents belonging to different experts' designs and the selection of one of these propositions. The main actors of MoTrans-BDI are the following :

- The User : provides the source metamodel (SMM), the target metamodel (TMM) and the SM to transform, as well as the QoS requirements.
- The MA : sends the fragments of the SM to BTAs, retrieves the proposed TMs and selects the final TM.
- The BTAs : after receiving an SM fragment, they produce TMs propositions and send them to the MA.

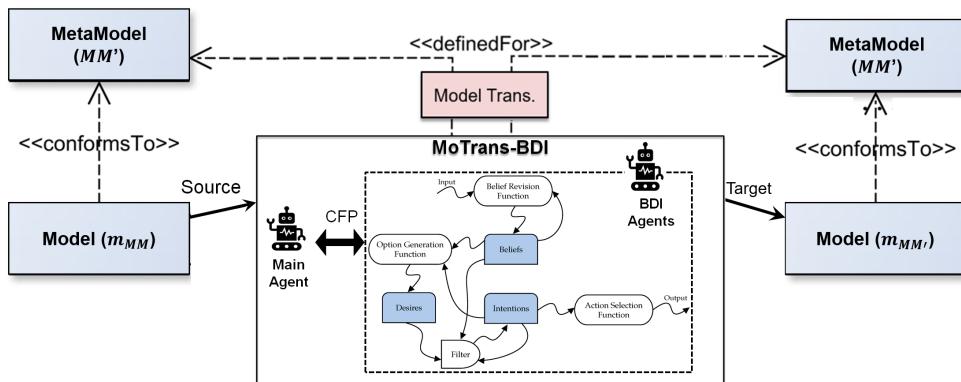


FIGURE 4.1 – Overview of MoTrans-BDI Framework

4.2.2 Theoretical Foundation of MoTrans-BDI Framework

The BDI agent architecture is based on Michael Bratman's philosophical theory [11] that explains reasoning through the following concepts : beliefs, desires and intentions. Consequently, all BTAs are autonomous agents that make use of these three concepts for their functioning. Let us define these concepts in the context of our framework.

Beliefs are the agent's model of the environment, basically what it believes to be true. This component of the BDI architecture is usually represented as a dataset of facts. In the context of model transformation by examples, we define a belief as follows :

Définition 15 (Beliefs)

A belief unit is a set of quadruples $(SMM_j, SM_i, TMM_k, TM_i)$ connecting a source model SM_i , belonging to a SMM_j , to its transformation target model TM_i belonging to a TMM_k . In other words, agent's beliefs are all model transformations it can accomplish. An agent may have a set of beliefs Bel defined as follows :

$$Bel = \{bel_1, bel_2, \dots, bel_n\} / bel_i = (SMM_j, SM_i, TMM_k, TM_i), \text{ With } 1 \leq i \leq n, 1 \leq j \leq m, 1 \leq k \leq t.$$

Where "n" is the total number of beliefs, "m" is the number of source metamodels the expert is able to transform, and "t" is the number of target metamodels defining the targets of transformations.

Desires represent the ideal state of the environment for the agent. Like in the human mind, these represent things we would like to see accomplished in the future. In our BDI agent model, Desires are defined as follows

Définition 16 (Desires)

The Desires represent the quadruples that are candidates to be selected for the considered transformation. All quadruples that have the SMM equal to the SMM provided for transformation and the TMM equal to the TMM provided for transformation. That is, this set of quadruples is a subset of the beliefs' set of quadruples. Let SMMs be the source metamodel and TMMs be the target metamodel, the desires are defined as follows : $Des(t) = \{des_1, des_2, \dots, des_p\} / p \leq n$, where t is a specific model transformation (e.g. UML2REL), and $des_i = (SMM_j, SM_i, TMM_k, TM_i) / SMM_i = SMM_s \text{ and } TMM_k = TMM_s$

Intentions represent a subset of desires that the agent has taken as goals to be accomplished. In our context, Intentions can be defined in this way :

Définition 17 (Intentions)

Intentions are quadruples that are selected to the final round after filtering the Desires, only quadruples that present a pre-established transformation similarity threshold are candidates to contribute to the final solution of the transformation. Formally :

Int = Filter($Des(t)$, sim), where :

- **sim** : is a pre-established transformation similarity threshold (i.e. the transformation similarity threshold is controlled by genetic primitives)
- **Filter** : is a function that filters all Desires " $Des(t)$ " to obtain a subset including all Desires that satisfy a transformation similarity threshold

Plans are used by BTAs to seek the suitable target model among all target models that can be derived from Intentions set.

Définition 18 (Plan)

A plan is a set of actions that are executed on all quadruples of the Intentions set to achieve the agent's goal. This set of actions is the specific algorithm used by the BTA to derive a set of candidate TMs. We can view the plan as one of the MTBE approaches mentioned in the related work. Although, in our case all BTAs use the same genetic algorithm to derive the best TM solution.

Définition 19 (Heuristic)

Let $\mathcal{H} = \{H_1, H_2, \dots, H_n\}$ be a set of heuristics used to refine a given TM, where each heuristic H_i is a tuple defined as $< c_i, E_i, TMM_i, cri_i >$ where c_i is a condition that must be checked and applied on element E_i (e.g. like attributes, classes, associations) of TM that conforms to TMM_i , in which a system need to be optimized. In order to refine the final model by the MA or BTAs, each heuristic may be related to a QoS criteria (cri_i) such as the performance and maintainability, etc. The benefits of using heuristics are related to some QoS criteria or improvements on some model transformation features.

Example of Heuristic : Let us consider an example of a heuristic $H_{exp} < nb(table) \leq threshold$, $table, REL, Response\ time >$. We note that $threshold$ is a parameter recommended by domain engineers. In order to refine the final model, each TM proposal will be checked with H_{exp} as follows : All TMs that have $nb(table) \leq threshold$ (predefined threshold) are candidate to be selected with respect to H_{exp} that expresses the response time optimization as a QoS criteria. That is, the TMs satisfying H_{exp} are supposed having a less number of linked tables that are involved to get the final result of the query. Indeed, these links are materialized as join operations which may consume a significant amount of time when executing user's queries. To sum up, MoTrans-BDI uses BDI agents, where Beliefs represent tuples connecting each source model SM_i to its transformation target model TM_i , Desires are tuples that are candidates to be selected for the considered transformation, and Intentions or goals are tuples that are selected to the transformation final round after filtering the Desires.

Finally, the model transformation problem to produce the best target model TM^{Best} is seen as a model transformation process driven by the QoS attributes as described below :

The Model Transformation process

Input :

- A Source Model (SM).
- A set of constraints (e.g. imposed by the designer or according to the application constraints).

Goal :

- Getting an efficient solution (TM) that satisfies at most the QoS attributes.

Output :

TM^{Best} : The best target model (TM).

4.2.3 Conceptual Organization of MoTrans-BDI Framework

As stated above, our Framework is based on a \mathcal{BDI} agent design that is composed of three main components (beliefs, intentions, and desires). As portrayed in Figure [Figure 4.2](#), the root element of MoTrans-BDI's meta-model is the MoTrans-BDI class from where starts the instantiation. This class represents the agent entity that combines the two types of agents, namely, BDI (i.e., BDITransformer)

agents and reactive agents (i.e., MainAgent). Each `BDITransformer` instance has at least one transformation goal, one plan, and uses the `searchAlgorithm` class instance to proceed to model transformation. It is commonly composed of an instance of the `Belief` class, an instance of the `Desire` class, and an instance of the `Intention` class. Both types of agent are able to refine the TM thanks to the heuristic class. On the other hand, each `Belief` consists of two different parts of data : source models and target models.

figures/MetaModelBDI2.png

FIGURE 4.2 – Conceptual Organization of MoTrans-BDI Framework

In order to plug in a set of heuristics to MoTrans-BDI, we propose a Domain-Specific Language Design (DSL), called `HeuristicDL`, as depicted in Figure 4.3 that allows the designer to add these heuristics. In order to take into account several types of model transformation, we include all heuristic's components in `HeuristicDL` to allow to express any instance of heuristic related to the vocabulary of SM's and TM's concepts. Each instance must conform to `HeuristicDL` to be persisted and reused by BTA agents. An example of `HeuristicDL`'s instance may appear as the following.

Preconditions : In UML2REL transformation case, checking if the number of the root class aggregates (e.g. class "Date") in the TM is ≤ 3 (e.g. class "Quarter", class "Month" and class "Day") **AND** checking if the QoS attributes needed are *response time OR energy consumption*.

Action : In this case, *the TM having a number of root class aggregates ≤ 3 is selected.*

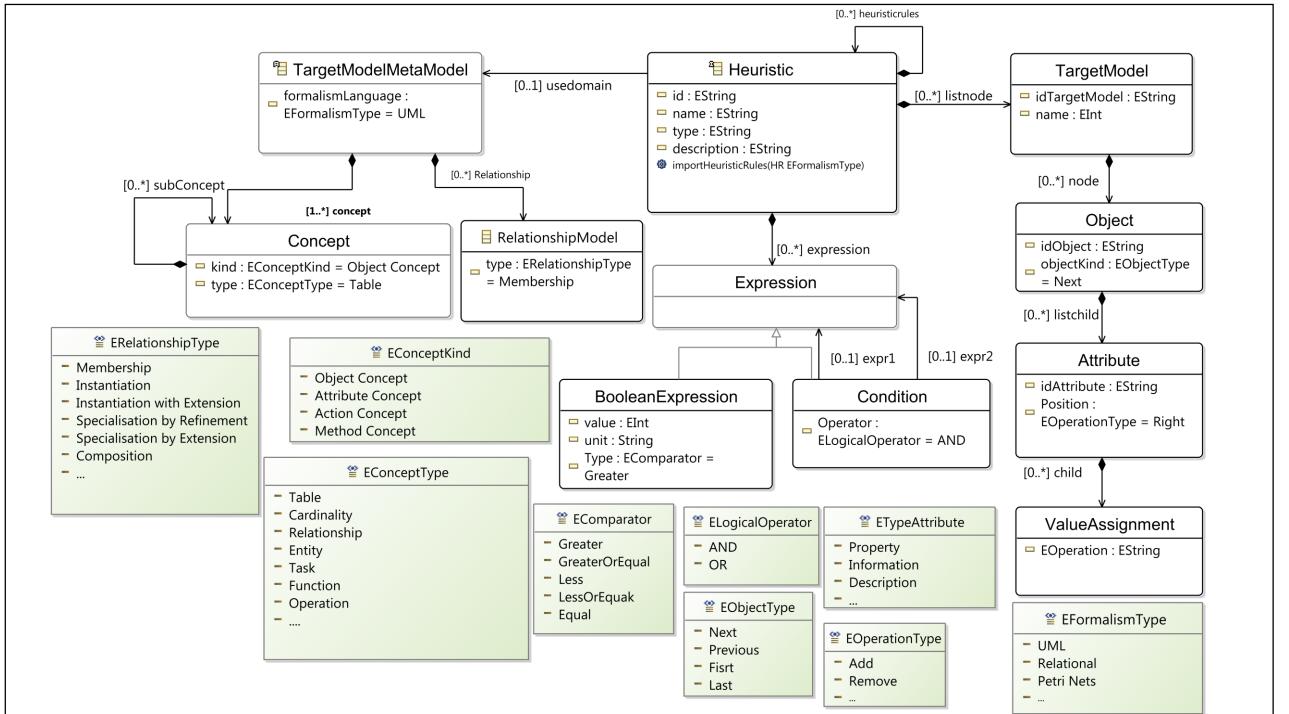


FIGURE 4.3 – Excerpt of our design language of model transformation heuristics.

An instance of a `Heuristic` class is made explicit by using as input the concepts related to the source-metamodel/target-metamodel that are illustrated by the `SourceElement` class and the `TargetElement`

class. A Heuristic is expressed as a set of assumptions by a combination of arithmetic and Boolean conditions. These assumptions are checked to be classified in one of the following three categories trueValueAssumption, undefineValueAssumption and falseValueAssumption.

4.2.4 The MoTrans-BDI Process

The model transformation process in MoTrans-BDI is structured as depicted in Figure Figure 4.4. It is composed of the four following steps :

1. Sending a call for proposal (CFP) by the MA to all BTAs;
2. Transformation processing and sending of proposals;
3. Refinement and selection of the best solution (the best target model);
4. Agent learning ;

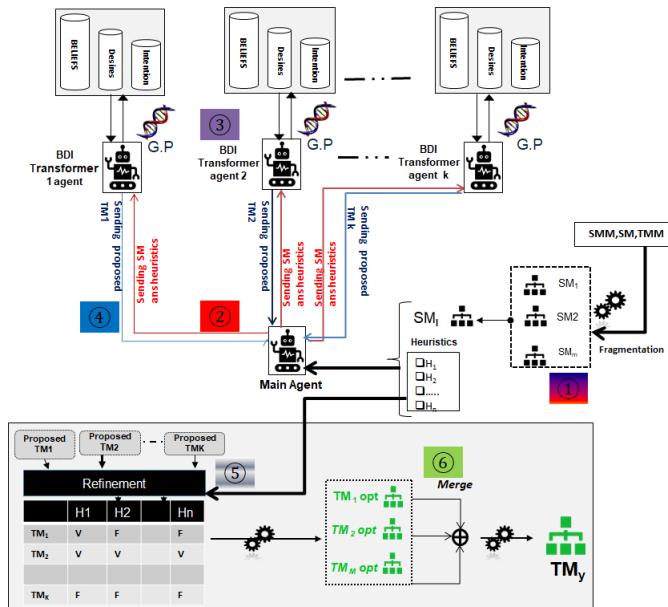


FIGURE 4.4 – The MoTrans-BDI model transformation process.

4.2.4.1 Sending a call for proposal

As depicted in Figure Figure 4.4, the MA picks out the SM to be transformed from the source models database and sends a triplet, composed of the SM metamodel, the SM and the TM metamodel (SMM, SM, TMM), accompanied by a set of heuristics to all BTAs. In the Contract Net Protocol architecture, this call is known as a CFP. The SMM and the TMM are used (by the BTA) to identify all quadruple examples that match the required model transformation.

4.2.4.2 Model transformation processing and sending of proposals

Considering both the examples and the QoS heuristics, the selection of the best TM by a BTA is a hard problem. Thus, we use a a genetic algorithm based two-steps process to this end. The first step

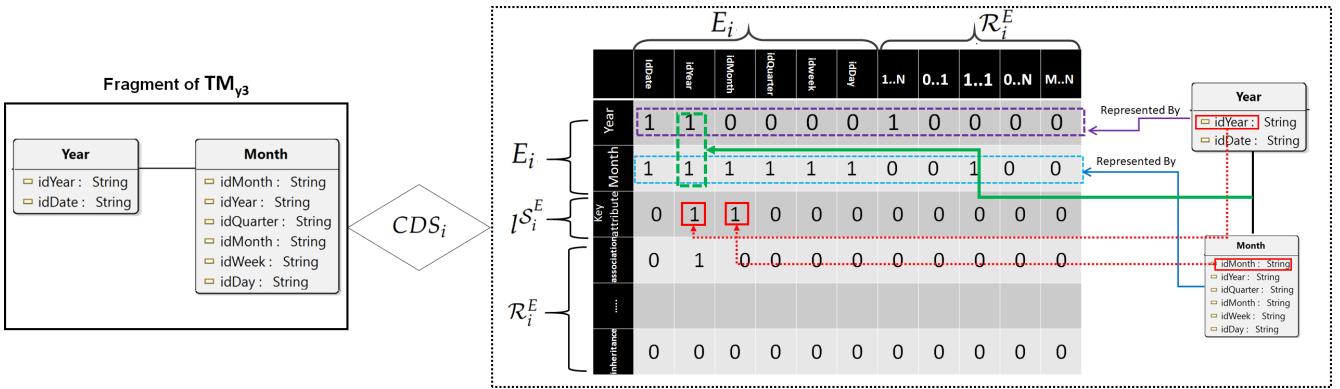
consists of generating the first (SM_i, TM_i) population where each SM_i may be equal or similar to the SM to be transformed. Then, starting from this population of (SM_i, TM_i), we apply the remain steps of the genetic algorithm [Algorithm 4.1](#) to derive the best TM guided by the QoS heuristics. The overall view of the genetic algorithm sequence is shown in Figure [Figure 4.8](#). In the following, we detail the TM selection process achieved by each BTA.

Beliefs filtering : after receiving the triplet (SMM, SM, TMM) from the MA, the first task achieved by the BTA consists of filtering the beliefs database so as to select only quadruplets having the SMM equals to the triplet's SMM and the TMM equals the triplet's TMM. This first subset represents the Desires of the BTA. All selected examples are concerned by the model transformations from SMM to TMM.

Desires filtering : a similarity algorithm is executed to select quadruples that present a pre-established transformation similarity threshold. In our case, we apply the EMF comparison tool to compare similarity between the source model to be transformed and the SM of each quadruple. The obtained examples represent the BTA's Intentions. These examples are then considered for the derivation of the best TM by applying the genetic algorithm with the heuristics.

Evolutionary transformation Phase : As the users model transformation requirements and QoS attributes evolve according to the user needs and the application constraints, we need an evolutionary solution that may take over this evolution by reproducing the recommended TM. consequently, the final solution (i.e. The best TM) will arise from the confrontation between different model pairs (SM,TM), we need an algorithm that started from the principle of combining the initial solutions to obtain the best solution. Our approach uses a Genetic Algorithm to find a set of TMs that maximize heuristics satisfaction and minimize the number of conformance errors with the TMM. The genetic algorithm starts from the population of TMs of the previous step. At each iteration, it computes an overall fitness value for each TM in the population and applies genetic operations until a stop criterion is fulfilled, finally, it returns the TM with the highest overall fitness. In the following, we describe all tasks needed to perform the model transformation by each BTA.

- **Coding principle :** The genetic algorithm requires first the TMs to be encoded in form of a feature vector that is able to evolve. Formally, we represent the coding of a given TM_i as a structure of Coding Sequence : $CDS_i = \langle E_i, \mathcal{S}_i^E, \mathcal{R}_i^E, l\mathcal{S}_i^E \rangle$, where we identify three subsets of objects $E = \{Class, Attribute, Association\}$ corresponding to the modifiable objects of the TM (i.e. objects that can be added, removed, and replaced). These objects are divided into properties/sub-properties denoted by $\mathcal{S}_i^{E_i}$ (e.g. multiplicities constraints of association). The relation \mathcal{R}_i^E represents the relationship (i.e. association, aggregation, composition, dependency and inheritance.) between elements E ($l\mathcal{R}^E : E_i \rightarrow E_j, E_i, E_j \in ExE$), e.g. class B is a subclass of class A. The label function $l\mathcal{S}_i^E$ assigns a unique string label to each sub-property $\mathcal{S}_i^{E_i}$, for example we can encode "the type of an attribute" by value "1" if it is a Key attribute and value "0" otherwise. Additionally the relation \mathcal{R}_i^E must keep the model transformation traces that produce the TM_i from the given SM_i . Finally, our encoding idea enables to generate the CDS of each TM_i solution according to its TMM . Therefore, the method used to code a TM solution must be adapted to each DSL. Indeed, our framework serves as recommending suggestions to the developer who needs to manually refine the coding to be specific to the DSL at hand. Figure [Figure 4.7](#) shows an example CDS_{y3} corresponding to a fragment of TM_{y3} .


 FIGURE 4.5 – Example CDS_{y3} corresponding to a fragment of TM_{y3} .

- **Generating the Initial Target Models Population :** Our approach uses the CDS in order to generate the initial population. The generation of the candidates TMs is based on the similarity between the input SM_i and the $top-k$ similar SMs in the dataset using the model comparison (M_i) or the domain specific distance functions proposed in the literature such as EMF Compare⁹, DSL Distance Measures [37] and DSDiffMM [41].

$$Similarity(SM_x, SM_i) = 1 - Distance_{M_i}(SM_x, SM_i) \quad (4.1)$$

The generation of the initial population is given by the CDS of all candidates TMs corresponding to the similar SMs .

The *Similarity* (SM_x, SM_i) is defined as a function to compare the similarity between all SMs on the basis of each SM 's CDS . The similarity is calculated based on binary data distance, it is represented by a percent proportion (see Equation Équation (4.3)).

$$\begin{aligned} Distance(CDS_i, CDS_j) &= Distance(E_i, E_j) + \\ &Distance(\mathcal{S}_i^E, \mathcal{S}_j^E) + Distance(\mathcal{R}_i^E, \mathcal{R}_j^E) + \\ &Distance(lS_i^E, lS_j^E) \quad (4.2) \end{aligned}$$

The distance between each pair of components of the CDS is calculated based on the similarity coefficients for Binary Data such as *Russel and Rao coefficient*, which is simply a proportion of positive matches [33] (see Equation Équation (4.3)).

$$Distance(O_i, O_j) = \frac{a}{a + b + c + d} \quad (4.3)$$

The values a, b, c and d are calculated using the *contingency table*¹⁰, with (resp. a, b, c, d) = number of positions where i is at (resp. "1", "1","0","0") and j is at (resp. "1", "0", "1", "0"), that means the number of times the two objects have the same positive occurrences.

9. <https://www.eclipse.org/emf/compare/>

10. The contingency table is a table showing the distribution of one variable in rows and another in columns, used to study the correlation between the two variables.

Exemple 1. Similarity calculate Let consider two objects O_i and O_j , we need to calculate the contingency table for : $O_i = (1, 0, 1, 1, 1)$ and $O_j = (1, 0, 1, 0, 0)$. The contingency table is calculated as illustrated in Figure Figure 4.6. By using the Russel and Rao coefficient formula to calculate the distance between the two objects : $a=2$, $b=2$, $c=0$, $d=1$, $\text{Distance}(O_i, O_j) = 0,4$.

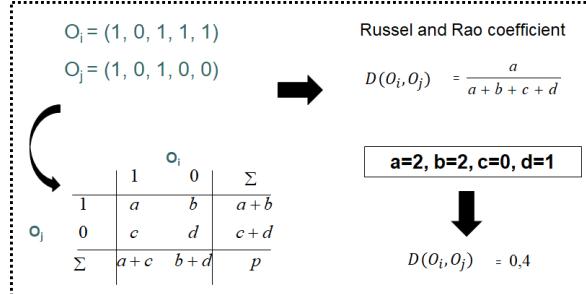


FIGURE 4.6 – Example of similarity coefficients for Binary Data

– **Genetic Operators :** The Genetic Algorithm is based on the following primitives : *crossover*, *mutation*, and *selection*. When a mutation and a crossover operator are applied, the goal is to slightly change the solution for the purpose to probably improve its fitness function (see Equation Équation (4.4)). The user who represents the model transformation designer can specify the probability thresholds for mutation and crossover (e.g. crossing and mutation rates). When a detected solution is not admissible, we remove one of the redundant model elements from the solution (i.e. vector populations). The genetic operators produce three main kinds of actions on the CDS, according to the DSL like adding, deleting, and modifying elements (e.g. add class, delete association between two objects, modify attributes' values). In addition, these genetic operators can replace an element by another that is equivalent but with a different type, they can also split or merge several models of the same type into a single one. Figure Figure 4.7 shows an example of multi-point crossover mechanism between two parents that correspond to two different TMs.

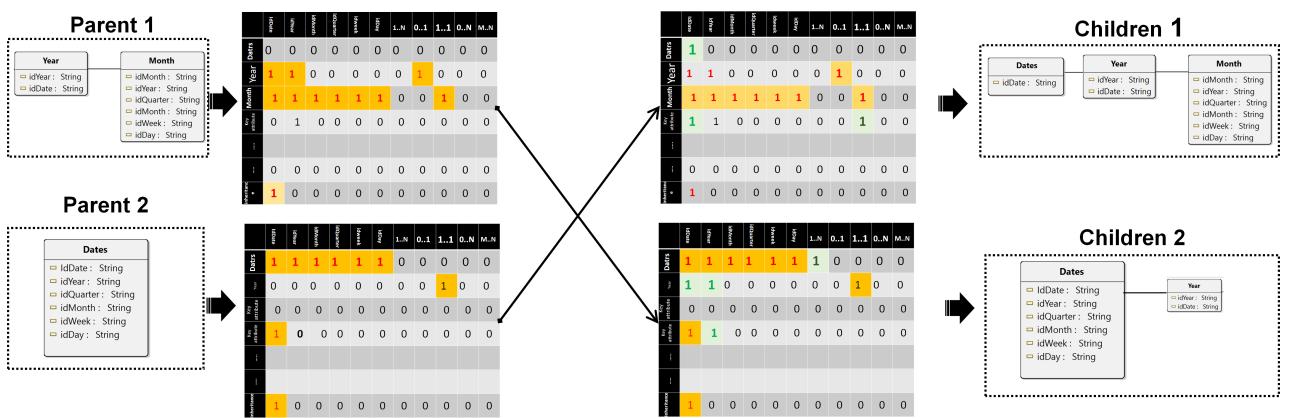


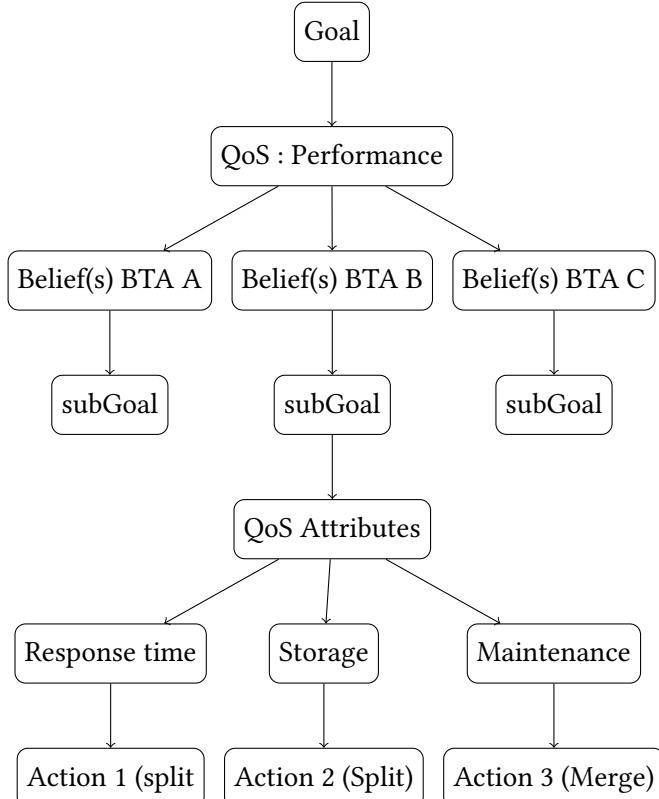
FIGURE 4.7 – Example of crossover of two parents (chromosomes) corresponding to two TMs

For each new generation, we must ensure the conformance of the reproduced instances against the TM metamodel. The choose of actions when a mutation and a crossover operator are applied by BTAs is driven by the transformation goal to finding best solution by satisfying in each

generation of our genetic algorithm more number of heuristics. Each heuristic may be related to improving of the QoS attributes such as, the performance and maintainability etc. Figure Section 4.2.4.2 shows an example of a goal hierarchy with an agent's goals, beliefs and actions. There is an initial goal where an initial task is broken down into sub-goals or sub-tasks and which is distributed among several agents ("top-down" problem decomposition).

Goal-hierarchy-of BTA-agent.

Goal-hierarchy-of BTA-agent



- **Fitness Function :** The fitness function, as defined by Equation Équation (4.4), represents the number of satisfied heuristics by a given TM_x whereas the constraints are handled by using the concept of the penalty function ($\text{Penalty}(TM_x)$), which penalizes infeasible solutions that violate the constraint of conformance against the TMM (see Equation Équation (4.5)).

$$Fitness(TM_x) = Nb(\mathcal{H} < H_1, H_2, \dots, H_n >) \models TM_x \quad (4.4)$$

$$\text{Penalty}(TM_x) = NbErrorOf(\neg \text{conforms } TMM) \quad (4.5)$$

To take into account the constraint of conformance error during the evolutionary transformation phase, we formalize the model transformation problem as a maximization objective function called $F(TM_x)$, defined from the $Fitness(TM_x)$ and $\text{Penalty}(TM_x)$, according to one of the three common modes : subtraction mode, division mode or subtraction/division mode. For example, the subtraction mode can be used as illustrated in equation Équation (4.6) :

$$F(TM_x) = \begin{cases} Fitness(TM_x) - \text{Penalty}(TM_x) \\ \quad \text{if } Fitness(TM_x) > \text{Penalty}(TM_x) \\ 0 \quad \text{else} \end{cases} \quad (4.6)$$

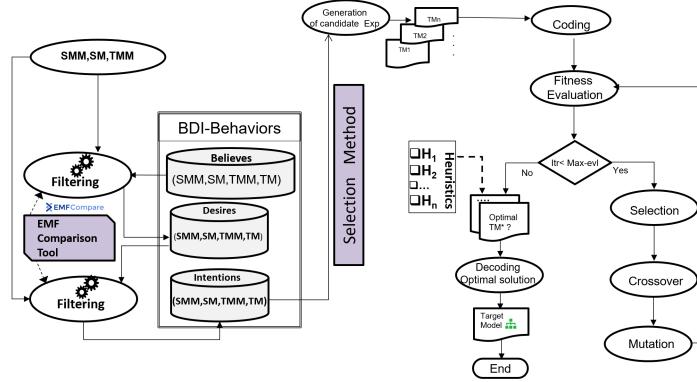


FIGURE 4.8 – The BTA model transformation process

4.2.4.3 Refinement of the proposed target models

When the MA receives the proposed TMs, generated by the evolutionary transformation phase, from the BTAs, we can distinguish three possible scenarios : (a) same decisions, (b) partial transformation, and (c) decisions with conflict (see Figure 4.9).

- In the scenario of "same decisions", the MA recommends the same decision without using any heuristic.
- In the scenario of "partial transformation", the MA recommends a TM as a model merging from the BTAs' different proposals.
- In the scenario of "Decisions with conflict" the MA uses the proposed algorithm (See Algorithm Algorithme 4.1) to refine the best solution.

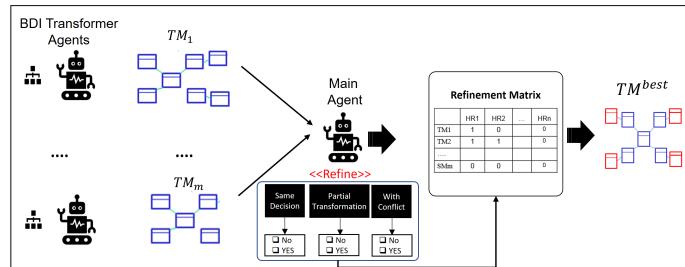


FIGURE 4.9 – Refinement scenarios illustration.

In order to refine the propositions provided by the BTAs, we define a set of heuristics to be used by the MA in order to recommend the best target model TM^{Best} . After receiving the proposed TMs that may represent the possible solution, the MA checks for each TM_i if the conditions e_k related to a given heuristic H_j are satisfied. This analysis produces three possible results : true, false or undefined (C^{undef} , C^{true} , C^{false}). TM^{Best} is the TM_i that satisfies the highest number of heuristics.

4.2.4.4 Agent Learning

The result returned by the MA after the TM refinement is considered as a new learned example in the beliefs' data : $bel_{new} = (SMM_j, SM_i, TMM_k, TM^{Best})$. This latter is sent to the BTAs, that do not

Algorithme 4.1 TM Refinement Algorithm

```

Entrée : The set of target models  $\mathcal{TM} = \{TM_1, TM_2, \dots, TM_n\}$                                 ▷ vector generated by BTAs
Entrée : The set of Heuristics  $\mathcal{H} = \{H_1, H_2, \dots, H_p\}$ 
Entrée : The set of conditions related to  $\mathcal{H}$ ,  $e = \{e_1, e_2, \dots, e_m\}$ 
Sortie : The Refined Target Model :  $TM^{Best} = \{ TM_i \in \mathcal{TM} \}$                                 ▷ i.e. a Target Model satisfying the conditions of Heuristics
1 :  $C = C^{undef} = C^{true} = C^{false} = \{\emptyset\}$ 
2 : pour target model  $TM_i \in \mathcal{TM}$  faire
3 :   pour Heuristic  $H_j \in \mathcal{H}$  faire
4 :     pour condition  $e_k$  of  $H_j$  faire
5 :       si  $TM_i \models e_k$  of  $H_j$  alors
6 :          $C^{true} \leftarrow C^{true} \cup \{e_k\}$ 
7 :       sinon
8 :         si  $TM_i \not\models e_k$  of  $H_j$  alors
9 :            $C^{false} \leftarrow C^{false} \cup \{e_k\}$ 
10 :        sinon
11 :           $C^{undef} \leftarrow C^{undef} \cup \{e_k\}$ 
12 :        fin si
13 :      fin si
14 :    fin pour
15 :  fin pour
16 : fin pour
17 :  $C \leftarrow C^{undef} \cup C^{true} \cup C^{false}$                                               ▷ Return the TM according that maximize heuristics satisfaction ( $\mathcal{H}$ )
    retourner  $TM$ ;
18 :

```

proposed the best TM, to contribute solving later new model transformation problems by injecting this new example in the initial population of the genetic algorithm. The examples can therefore also be seen as integrity constraints, the satisfaction of which is forced when adding a new example. Agent learning BTA is noted : $\mathcal{L} : \mathcal{D} \oplus_e bel_{new}$, \oplus_e is the belief update operators which means that the BTA agent learns a new knowledge under a constraint e , where is new example bel_{new} that be added on the dataset \mathcal{D} depending on the properties it satisfies e (i.e. no contradiction with bel_{new} and the existing pairs SM, TM in \mathcal{D}). For instance, a properties e : check if a new example bel_{new} of a one to many (1 :M) relationship type is in contradiction or not with existing BTA believes of (1 :M) relationship.

4.3 Conclusion



Adaptation de l'Algorithme Génétique pour le problème de reproduction des exemples de transformation des modèles

« Chaque individu apporte au monde sa contribution unique. »

— Jack Kornfield

Sommaire

5.1 Introduction	54
----------------------------	----

5.1 Introduction

Adaptation de l’Algorithme Génétique pour la Transformation des Modèles

6.1 Introduction

Dans ce chapitre, Nous commençons par démontrer la raison de notre sélection en examinant les approches de transformation existantes. Ensuite, nous passerons en revue la mise en œuvre de cette nouvelle stratégie, y compris les spécifications de l’algorithme.

6.2 Problèmes de transformation des modèles

Dans cette section, nous allons présenter la formalisation du problème de la transformation des modèles.

6.2.1 Formalisation du problème

Dans le cas d’un problème de transformation de modèle, le meilleur scénario pour transformer un modèle MSi source en un modèle cible MCj idéal répondant aux besoins de l’utilisateur et/ou de l’application et répondant aux l’objectif de la transformation, le problème est l’explosion du nombre des instances de modèles cibles qui dépend du nombre de fragments générés à partir de modèle source (MS). Par exemple dans le cas de transformation d’un modèle relationnel ER vers un modèle cible fragmenté qui répond du problème de l’optimisation des requêtes SQL. Si le nombre de fragment de table T_i est m_i , alors le nombre de fragments de la table qui contient des clés étrangères est $N = \prod m_i$.

Le problème de sélection du schéma de fragmentation qui est le modèle cible MC , en considérant une contrainte sur le nombre de fragments généré, a été formalisé comme suit :

Etant donné :

- $Q = \{Q_1, \dots, Q_m\}$ un ensemble de requêtes.
- \mathcal{R} une table qui contient des clés étrangères et $\mathcal{T} = \{t_1, \dots, t_n\}$, t tables de modèle MS .
- W le nombre de fragment de la table de relation \mathcal{R} imposé par le concepteur
- Le problème se sélection d’un MC en se basant les opération de transformation : fragmenter et fusionner consiste à fragmenter et/ou fusionner la table de \mathcal{R} en N fragments selon d’un schema de fragmentation correspond au MS des tables \mathcal{T} tel que :
 - Le coût d’exécution des requêtes soit minimal
 - Le nombre de fragment de F soit minimisé : $N \leq W$

6.2.2 Démarche de transformation

La sélection d'un modèle cible optimal basé sur l'opérateur split/merge (Split fragmenter Merge pour fusionner) quatre étapes importantes : d'abord les informations qui permettent de la spécification d'un schéma de transformation sont extraites. Ensuite, un codage d'un schéma de transformation est spécifié. Puis un ensemble de schéma candidats est générer. Enfin, parmi cet ensemble de schémas possibles, le schéma de fragmentation optimal est sélectionné.

- **Préparer la transformation** : A partir de la charge de requêtes, les prédictats de sélection sont extraits. Ces prédictats permettent de spécifier les attributs de sélection sur lesquels portera la transformation des tables T et dont les domaines sont découps en sous domaines. Ces attributs permettent, à leur tour, d'identifier les dimensions candidates à la transformation target/fusionnement. Puis, pour chaque table de $MS T$, un ensemble de prédictats complet et minimal est générer.
- **Coder un modèle Source à transformé** : Chaque attribut de transformation AR_i de la table T_k , extrait au niveau de l'étape précédente, possède un domaine de valeurs t_i . Le découpage du domaine t_i en n sous domaine st_1, \dots, st_{n_i} permet de spécifier n_i fragments de la table T_k . Pour t attributs AF_1, \dots, AF_t de la table T_k le nombre de fragments total est égale à $\prod n_i$.

Ce nombre représente le nombre de fragments maximum pour une table, il se peut que certains domaines soient regrouper dans le même domaine ce qui permet de générer un nombre des fragments T inférieur à $\prod n_i$. Afin d'exprimer cela, les auteurs proposent un codage du schéma cible de transformation qui attribue à chaque sous domaine st_j de chaque AR_i un numéro. Les sous domaines du même AF_i ayant le même numéro définissent le même fragment.

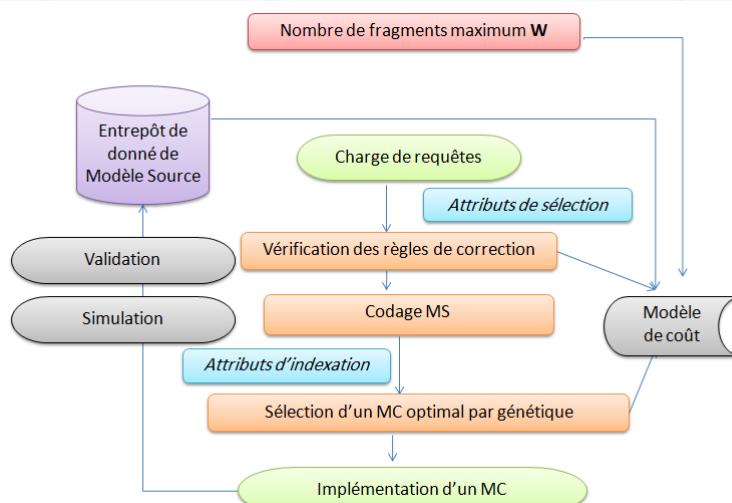


FIGURE 6.1 – Démarche des modèles de transformation

Exemple 01

Soit un modèle source dont le schéma est donné par la figure 2. Soient les attributs de fragmentation suivants : "Genre" avec les valeurs {F, M}, "Classe" avec les valeurs {A, B, C} et "Age" dont les valeurs sont représentées par l'intervalle [0, 60]. Le découpage en sous domaines des domaines de ces attributs

est : Genre : {F} et {M}, Classe : {A}, {B} et {C}, enfin Age = [0, 20[, [20, 40[et [40, 60].



FIGURE 6.2 – Figure 2 :Schéma d'un modèle source

Un codage du schéma de transformation est représenté codage d'un shéma de transformation .

Ce codage présenté sur la figure 3 montre les points suivants :

1. Les tables concernées par la fragmentation sont la table *Client* sur les attributs *Age* et *Genre* et la table *Produit* sur l'attribut *Classe*.
2. Pour la table Client, chaque sous domaine de l'attribut possède un numéro différent. Par contre, les domaines de l'attribut *Genre* possèdent le même numéro. Cet attribut ne participe donc pas au processus de fragmentation. Par consequent la Table Client est fragmentée en trois fragments selon l'attribut *Age*.
3. Pour la dimension Produit, les domaines {A} et {B} de *Classe* sont regroupés en un seul domaine (ils possèdent le même numéro 0), ce qui signifie que les tuples Produit dont la *Classe* est égale à "A" ou "B" appartiennent au même fragment. Cela donne deux fragments de la table *Produit*.
4. **Générer une solution initiale :** les auteurs proposent d'utiliser l'algorithme d'affinités adapté à la FH dans les entrepôts groupement ne se fait pas sur les prédictats mais sur les sous domaines des attributs. Une affinité entre deux sous-domaines est la somme des fréquences d'accès des requêtes accédant simultanément aux deux sous-domaines. Appliquer une heuristique : La solution initiale est améliorée grâce à l'application des heuristiques.



FIGURE 6.3 – Figure 3 :Codage d'un schéma de Transformation

a) Réécriture des requêtes

Les auteurs ont proposé une nouvelle démarche de fragmentation non supportée par les SGBD commerciaux. Pour les bénéficiers, il faut réécrire les requêtes sur le nouveau schéma de données. Cette réécriture a pour but de charger uniquement les fragments nécessaires pour l'exécution d'une requête donnée. Une fois l'entrepôt de données fragmenté, il peut être représenté sous forme de plusieurs sous schémas en étoile. Ainsi, pour calculer le coût d'exécution d'une requête sur un schéma fragmenté, il faut calculer ce coût sur chaque sous schéma. Avant cela, il faut identifier les sous schémas valides et leur correspondance pour une requête.

1. Identifier les sous schéma de MC valides pour la requête comme suit :
2. Identifier les fragments de dimensions valides pour la requête (comparer les prédictats de la requête et ceux formant la clause définissant chaque fragment de dimension)
3. Utiliser ces fragments pour déterminer les fragments de faits valides.
4. Exécuter la requête sur chaque sous schéma. Pour ce faire il faut identifier la correspondance entre la requête et le fragment de faits pour déterminer les jointures à exécuter en plus des jointures spécifiées dans la clause WHERE. Cette correspondance peut être :
 - **Totale** : la réponse à la requête est le fragment de faits, aucune jointure n'est requise
 - **Partielle** : le fragment de faits contient des tuples non pertinents pour la requête, il faut effectuer des jointures avec les tables de dimensions pour les éliminer.

6.2.3 Modèle de coût

La transformation de modèle est effectuée par heuristique à savoir l'algorithme génétique. Cet algorithme est guidé par un modèle de coût qui permet d'évaluer chaque schéma de transformation cible généré par génétique. Cela permet de choisir la configuration qui optimisent le mieux la charge de requête. En prenant l'exemple de la fragmentation, les approches qui se basent sur la complétude et minimalité des prédicts ou ceux qui utilisent les affinités ne donnent aucun mécanisme permettant d'estimer la qualité du schéma de fragmentation obtenu. Pour les algorithmes à base d'affinités par exemple, le paramètre essentiel est la fréquence d'accès des requêtes aux données. Ce paramètre n'est pas suffisant car certaines requêtes non fréquentes peuvent engendrer un coût d'exécution important. Il faut utiliser d'autres paramètres comme la taille des tables, la taille des tuples, les facteurs de sélectivité des prédicts de sélection et la taille d'une page système. De plus, le choix d'un schéma de fragmentation est un problème NP-Complet. Ils nécessitent l'utilisation d'heuristiques qui se basent sur une fonction objectif. Cette fonction est spécifiée à partir du modèle de coût.

Le modèle de coût permet d'estimer le coût d'exécution des requêtes. Il est composé de deux grandeurs :

(1) le nombre d'opération d'entrée sortie (E/S) nécessaire pour charger les données exploité par une requête.

(2) le coût de calcul CPU. Généralement, le coût CPU est négligeable devant le coût d'E/S. Ainsi, seul ce dernier est pris en compte pour estimer le coût d'exécution.

Dans notre travail, nous allons effectuer la transformation de modèle

à base d'algorithme génétique. Il faut donc établir un modèle de coût mathématique qui va permettre de spécifier la fonction objectif du génétique et guider la sélection des structures d'optimisation. Afin d'établir ce modèle de coût, nous présentons la formulation du coût d'exécution des jointures ainsi que le coût d'exécution d'une requête sur un schéma non optimisé.

b) Formule de coût sur un schéma source non optimisé Soit un schéma source d'un modèle de données modélisé en étoile avec une table de fait R et d tables de dimensions T1,...Td. Soit une charge de requête Q1,...Qn. Le coût d'exécution d'une requête sur un entrepôt de données représente le coût de chargement des tables pour effectuer les jointures par hachage, il est donné par la formule suivante (on suppose que les jointures intermédiaires ne nécessitent pas une écriture sur disque) :

$$Cost(Q_i) = 3 \times (|R| + \sum d |T_i|)$$

Nous allons présenter, dans les sections qui suivent, la formulation du modèle de coût pour la fragmentation horizontale et les index de jointures binaires.

6.2.4 Modèle de coût pour la transformation

Le modèle de coût, que nous allons utiliser pour évaluer un schéma source et le schéma de transformation cible (fragmenté), s'inspire du modèle avancé dans [1]. Ce modèle permet de calculer le nombre

d’E/S nécessaires pour exécuter une charge de requêtes sur un schéma fragmenté par fragmentation horizontale. Plus exactement une fragmentation horizontale primaire sur les dimensions et une dérivée sur la table de fait. Chaque requête comporte des prédictats de selection et des jointures en étoile entre fait et dimension.

Les paramètres utilisés dans ce modèle de coût sont classés en trois catégories :

Les paramètres sur l’entrepôt de données comme la taille des tables, la taille d’un tuple, la taille des tables en pages systèmes nécessaires pour leur stockage.

Les paramètres sur le système physique à savoir la taille de la page système.

Les paramètres sur la charge de requêtes comme la sélectivité des prédictats.

Nous considérons un entrepôt de données modélisé en étoile avec une table de fait R et d table de dimensions T1,...Td. La fragmentation de l’entrepôt donne N sous schéma en étoiles S1, ...SN. Soit une requête Q à exécuter sur l’entrepôt.

Le coût d’exécution d’une requête sur un sous schéma Si estime

Le coût d’exécution d’une requête sur un sous schéma Si estime le coût de chargement du fragment fait puis la jointure entre ce fragment et les fragments dimensions.

$$Cost(Q, MS_i) = valide(Q, S_i) \times \left[3 \times \left[\prod_{j=1}^{mi} Sel(PR_j) \times |R| \sum_{k=1}^d \prod_{j=1}^{L_i} sel(PM_j) \times |T_k| \right] \right] \quad (6.1)$$

Le cout total d’exécution de la requete sur tout l’entrepot fragmenté est

$$Cost(Q, SR) = \sum_{i=1}^N Cost(Q, S_i)$$

6.2.5 Fonction objectif pour l’algorithme génétique

Afin de sélectionner les MC nous avons utilisé un algorithme génétique qui exploite une fonction objectif. Cette fonction évalue, pour chaque itération de l’algorithme génétique, le taux d’optimisation du coût d’exécution, en termes d’E/S, d’un ensemble de requête sur un modèle sible .Cette évaluation est effectuée en comparant ce coût par rapport au coût d’exécution des requêtes sur un entrepôt non optimisé. Ainsi, le but de l’algorithme génétique est de trouver la solution (schéma de fragmentation) qui minimise la fonction objectif. Nous présentons, dans ce qui suit, la formulation générale d’une fonction objectif ainsi que les différents types de fonctions. Nous exposons ensuite la fonction objectif pour la sélection d’un schéma de fragmentation et celle utilisée pour sélection des index

6.2.5.1 Formulation d’une fonction objectif

Soit $F(x)$ comme une fonction objectif utilisée dans une heuristique. On s’intéresse généralement à Maximiser $F(x)$ sous la contrainte $C(x)$. C’est un problème d’optimisation d’une function F sous une contrainte C. Soit P en (x) une function pénalité que penalise une solution x qui viole une contrainte. L’utilisation de la fonction de pénalité pour la fonction $F(x)$ permet de transformer le problème en un

problème d'optimisation sans contraintes d'une fonction $F(x)$ définie à partir des fonctions $F(x)$ et $Pen(x)$. en utilisant le modes division, , **division** [2].

Mode division :

$$F'(x) = \begin{cases} \frac{f(x)}{Pen(x)} & \text{si } Pen(x) \neq 0 \\ F(x) & \text{si non} \end{cases} \quad (6.2)$$

Dans nos travaux, nous avons choisi d'utilisé le mode division. Mais la formulation de la fonction objectif nécessite d'être adaptée à notre problème de sélection. En effet, le problème présenté ainsi est un problème de maximisation. Or, notre sélection de structure d'optimisation se base sur la minimisation du coût d'exécution des requêtes. L'adaptation du problème nécessite une reformulation de la fonction pénalité. Cette fonction doit pénaliser les solutions qui violent la contrainte en augmentant leur fonction objectif. Nous présentons dans ce qui, la fonction objectif choisie pour la sélection d'index et celle adoptee pour la sélection d'un schema de fragmentation.

On distingue trois types de fonction objectif permettant d'évaluer le coût d'exécution d'une charge de requête sur un MS non optimisé

- La première fonction « Fonction objectif profit » privilégie les structures d'optimisation qui apporte les plus de profit lors de l'exécution de la charge de requête.
- La seconde « Fonction objectif ratio profit/contrainte » favorise les structures d'optimisation qui apportent le plus de profit mais qui ne viole pas une contrainte C.
- Enfin la troisième fonction « Fonction objectif hybride » représente une hybridation des deux fonctions annoncée. Elle permet de sélectionner, en premier lieu, les structures qui apportent le plus de profit. Ensuite, si la contrainte risqué d'être violée, la fonction objectif sélectionne les structures qui apportent le plus de profit mais qui ne viole pas la contrainte.

Pour notre approche de sélection nous avons retenue la fonction objectif avec ratio profit/contrainte. Elle permet de sélection des modèles cibles optimales sous la contrainte liée à la technique, cela en spécifiant la function F et la function pénalité P en à partir de la contrainte ,puis determiner la function générale F'.

6.2.5.2 Fonction objectif pour la fragmentation

Nous allons présenter, dans ce qui suit, la formulation de la fonction objectif ainsi que la fonction pénalité pour la sélection, par algorithme génétique, d'un schéma de fragmentation.

Soit un entrepôt de données modélisé en étoile avec une table de fait F et d tables de dimensions D1,...Dd. nous allons considérer ce qui suit :

- Une charge de t requête $Q = Q_1, \dots, Q_t$ à executer sur un modèle source MS .
- L'espace de m solutions ou schémas de fragmentations possibles générés par l'algorithme génétique $SF = SF_1, \dots, SF_m$.
- N_{sf_i} sous schéma en étoiles relatifs à un schéma $SF_i = S_1, \dots, SN_i$. Rappelons le coût d'exécution d'une requête Q_k sur un sous schéma Sj :

$$Cost(Q_k, S_j) = Valide(Q_k, S_j) \times \left[3 \times \left[\prod_{p=1}^{M_j} Sel(P_{F_p}) \times |F| + \sum_{\substack{p=1 \\ S=1}}^d Sel(P_{M_p}) \times |D_s| \right] \right] \quad (6.3)$$

Avec M_j

le nombre de prédictat de sélection qui définissent un fragment fait L_j^S

le nombre de prédictat qui difinissent un fragmenté est le suivant :

$$Cost(Q_k, SF_i) = \sum_{j=1}^{N_S f_i} Cost(Q_t, SF_j) \quad (6.4)$$

On peut donc exprimer le cout d'exécution de tout la charge de requete sur l'entre pot fragmente :

$$Pen(SF_i) = \frac{N_S f_i}{W} \quad (6.5)$$

Afin de formule la fonction objectif, il faut définir la fonction pénalité , Etant donné une contrainte sur le nombre de fragments fait maximum W , l'algorithme génétique peut générer des solution SF_i dont le nombre de fragmentation est égale à :

$$Pen(SF_i) = \frac{N_S f_i}{W} \quad (6.6)$$

Enfin l'algorithme génétique doit sélectionné un shéma de fragmentation qui minimise la fonction objectif ration (*profit*) nombre de fragment maximiser suivant :

$$F'(SF_i) = \begin{cases} Cost(Q, SF_i) \times Pen(SF_i) & \text{si } Pen(SF_i) > 1 \\ Cost(Q, SF_i) & \text{si non} \end{cases} \quad (6.7)$$

6.2.6 Mécanisme de codage d'un schéma de transformation

Nous présentons dans cette section le mécanisme de codage que nous avons adopté pour représenter un schéma de transformation de modèle.

6.2.6.1 Génération d'un modèle de transformation à partir d'un codage

Tout algorithme exploitant notre mécanisme de codage, doit nécessairement pouvoir le décoder, c'est-à-dire générer le schéma de fragmentation correspondant à un code donné. Ce décodage se fait en deux étapes :

1. génération des schémas de fragmentation des tables de dimension,

2. génération du schéma de fragmentation de la table des faits.

6.2.6.2 Fonction Fitness

Un individu dans notre algorithme génétique représente une solution possible du problème donc un schéma de transformation (TM). Pour évaluer les différentes solutions, nous avons défini une fonction fitness basée sur le modèle de coût vu précédemment. Nous avons formalisé le problème de fragmentation horizontale comme un problème de minimisation, or les algorithmes génétiques traitent des problèmes de maximisation. Pour traiter un problème de minimisation avec les algorithmes génétiques, une transformation de la fonction à minimiser est souvent effectuée pour le ramener à un problème de maximisation. Par conséquent, nous proposons la maximisation du gain obtenu entre le coût d'exécution des requêtes sur l'entrepôt initiale et le coût totale des requêtes de la charge sur un schéma de fragmentation. Le gain peut être calculé comme suit : $\text{Gain} = \text{CoutInit} - \text{CoutSCH}$, où : - CoutInit : coût d'exécution des requêtes sur l'entrepôt initiale non fragmentée. - CoutSCH : coût d'exécution des requêtes sur un schéma de fragmentation. Notre AG peut générer des solutions violant la contrainte de maintenance (solutions inadmissibles). Pour ne pas être confronté à une population totalement inadmissible, nous nous assurant qu'au moins la moitié de la population est admissible, ceci est réalisé en réparant les solutions inadmissibles. Cette réparation consiste à fusionné des sous-domaines appartenant à des fragments différents d'un attribut choisis aléatoirement afin d'obtenir un nombre de sous-schémas qui satisfait la contrainte W fixée par le concepteur. Rappelons que notre algorithme génétique garde trace de la meilleure solution de chaque génération, de ce fait, nous nous assurons que le meilleur individu sélectionné d'une génération soit admissible, c.-à-d que si ce dernier est inadmissible, nous procédons à sa réparation puis à sa réévaluation, ensuite nous le comparons avec le meilleur individu admissible de sa génération, nous sélectionnons enfin le meilleur des deux comme étend le Best de cette génération. Afin de pénaliser ces schémas, une fonction de pénalité Pen(MS) est introduite et fait partie de la fonction objectif, elle est définie comme suit :

$$\text{Pen}(MS) = W / \text{nbFragTR}, \text{ où :}$$

- W : contrainte du nombre de fragments maximum à ne pas dépassé.
- nbFragSCH : nombre de fragments générés par un schéma de fragmentation.

La valeur de Pen est < 0 si le nombre de fragments d'un schéma est supérieur à W, ce qui diminue la valeur de la fonction objectif F(SM) et défavorise la probabilité de sélection.

La valeur de Pen est > 0 dans le cas contraire ce qui permet d'augmenter la valeur de F(SM) ainsi que la probabilité de sélection.

$$F(SF) = \text{Gain} * \text{Pen}(SM)$$

6.2.6.3 Génération de la population initiale

Pour procéder à la création de la population initiale, nous générions aléatoirement N solutions pour former les individus de cette première génération où N est un nombre constant fixé par l'administrateur et représente le nombre d'individus par génération. La génération aléatoire des solutions est effectuée en remplies pour chaque individu, d'une manière aléatoire, les cellules dans le tableau multidimensionnel.

La numérotation aléatoire des cellules présente un problème de multi-instanciation. Pour résoudre ce problème, nous appliquons la fonction Re Numéroter que nous avons définie auparavant. Cette fonction sera appliquée sur toutes les lignes de tous les individus de la population initiale. L’algorithme génétique améliore ensuite la population initiale en lui appliquant les opérateurs génétiques de sélection, croisement et mutation que nous décrivons dans les sections suivantes.

6.2.6.4 Sélection

L’opérateur de sélection permet de sélectionner les meilleurs individus d’une population pour participer dans la génération de la population suivante. Dans le cadre de notre travail, nous avons utilisé la sélection par roulette. Cette méthode associe chaque individu à sa qualité (valeur de la fonction d’évaluation).

La roulette est partagée entre tous les individus, la surface allouée à chaque individu est en fonction de la qualité de l’individu. Par conséquent, les meilleurs individus ont plus de chances d’être sélectionnés.

L’implémentation de cette méthode se fait de la manière suivante :

- Calculer la valeur de la fonction d’évaluation de chaque individu et la somme totale de ces valeurs.
- Chaque individu est associé à sa probabilité de sélection (segment de longueur égal à sa valeur de fitness divisée par la somme des fitness déjà calculée).
- Tirer un nombre aléatoire et choisir l’individu dont le segment en globe ce nombre.

6.2.6.5 Croisement

Le croisement est un opérateur de recombinaison qui permet à deux individus d’échanger leurs gènes en vue de créer de nouveaux individus plus intéressants. Le croisement est appliqué sur deux individus pères choisis par l’opérateur de sélection. Nous avons utilisé le croisement multipoints entre individus qui peut être considéré comme un croisement mono-point sur chaque ligne (attribut) de l’individu. Après le choix de la position de croisement sur chaque attribut, les deux individus échangent leurs gènes. Cela est effectué en échangeant les numéros des cellules qui se trouvent après le point de croisement entre les deux individus. Cet échange peut rendre la numérotation non conforme, ce qui nous amène à appliquer la fonction ReNuméroter sur tous les attributs des individus fils avant l’injection dans la population suivante (Voir Figure 8).

Le croisement est effectué avec un taux de croisement Tc, généralement grand pour pouvoir combiner des solutions et former des générations de meilleure qualité. Nous décrivons le fonctionnement de l’opérateur de croisement par les étapes suivantes :

- Tirer un nombre aléatoire N (entre 0 et 100)
- Si $N > Tc$ alors il n’y aura pas de croisement. Nous réinjectons les deux parents dans la population suivante.
- Si $N < Tc$ alors sur chaque ligne choisir aléatoirement une position de croisement et échanger les gènes entre les deux individus.
- ReNuméroter les deux individus et les injecter dans la population suivante.

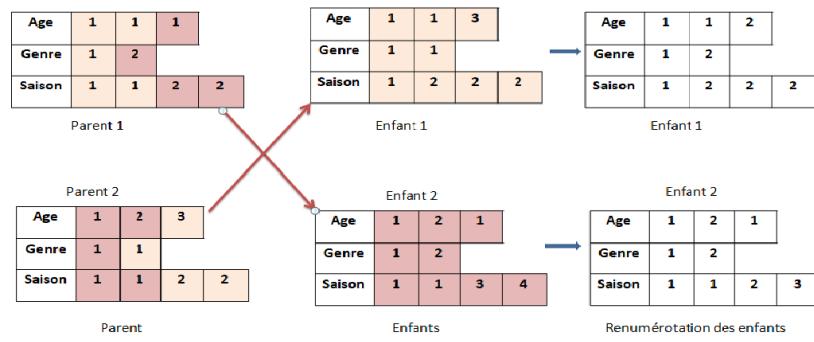


FIGURE 6.4 – Figure 9 :Exemple de croisement

6.2.6.6 Mutation

L'opérateur de mutation permet de modifier occasionnellement des gènes d'un individu pour permettre d'explorer certaines zones dans la codification des individus où le croisement ne peut pas explorer. La mutation se fait avec un taux T_m généralement petit. L'opérateur de mutation que nous avons utilisé permet de modifier une case du tableau multidimensionnel représentant l'individu choisi. Notons que la modification d'une case dans une ligne peut engendrer un problème de multi-instanciation. Pour résoudre ce problème, chaque ligne sera re-numérotée (Voir Figure10).

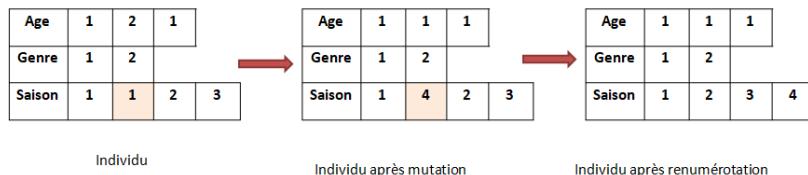


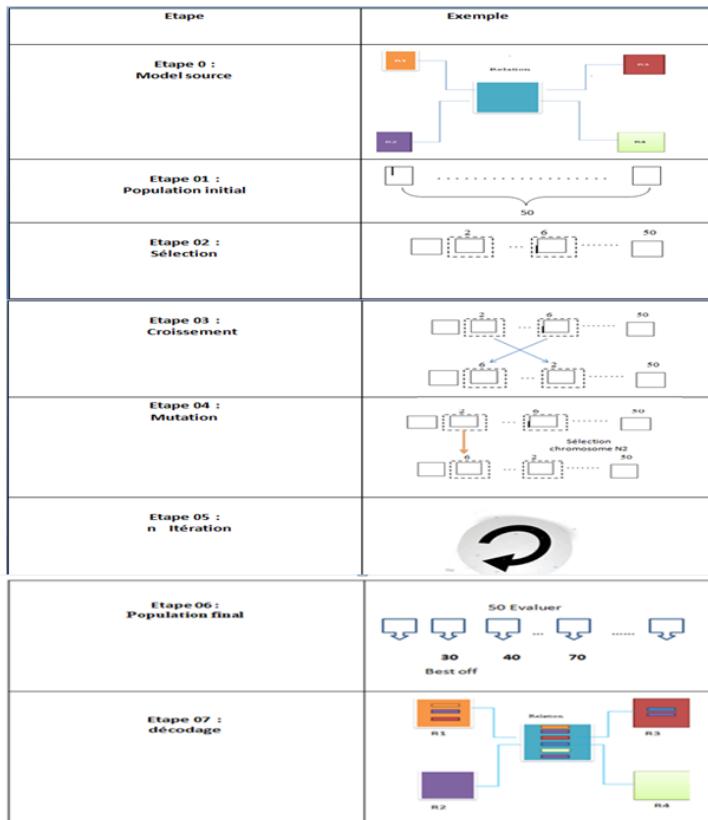
FIGURE 6.5 – Figure 10 :Exemple de mutation

Nous pouvons décrire le fonctionnement de l'opérateur de mutation par les étapes suivantes :

- Déterminer aléatoirement le nombre d'individus à muter
- Choisir aléatoirement ces individus.
- Tirer un nombre N aléatoire (entre 0 et 100)
- Si $N > T_m$ alors injecter l'individu dans la population suivante sans mutation.
- Sinon, pour chaque individu choisi, choisir aléatoirement une ligne
- Choisir aléatoirement sur cette ligne une case à muter
- Donner une valeur aléatoire à cette case (entre 1 et le nombre de cases).
- Renuméroter la ligne modifiée.
- Injecter l'individu résultat dans la population suivante.

Exemple d'instanciation

La figure suivante illustre le déroulement d'un exemple d'instcaition de notre approche proposée. Injecter l'individu résultat dans la population suivante



6.3 conclusion

Dans ce chapitre, nous avons proposé une nouvelle démarche de trasnformation des modèles en se basant sur le schéma de fragmentation. Cette démarche se base sur l'algorithme genetique. Ainsi, nous avons décrit l'architecture de notre démarche de transformation, nous avons explicité les algorithmes de transformation et le modèle de coût utilisé pour guider la sélection des schemas cibles



Outil Suppport de notre cadre de transformation MTBE

« A simple paradigm for nooconomics, the economy of knowledge. »

— Idriss J. Aberkane

Sommaire

7.1	Introduction	68
7.2	MoTrans-BDI Implementation and Evaluation	68
7.2.1	Implementation	68
7.2.2	Evaluation of MoTrans-BDI	70
7.2.3	Evaluation of MoTrans-BDI proposal's impact against Performance Measurement	74
7.3	Threats to validity	76

7.1 Introduction

chapitre5/chap5Fig/

7.2 MoTrans-BDI Implementation and Evaluation

We implemented our approach in a prototype tool. Then, we conducted an evaluation with an illustrative case study.

7.2.1 Implementation

Our approach has been implemented and made available as an open-source code. All artefacts are available with this submission as a zip file, as well as a Git repository¹¹. To ease the understanding, we provide a README file in the package that specifies how to use the code. The solution we adopted is as follows :

- The system is deployed in a centralized architecture, on a Windows Server 2019 virtual machine.
- BDI agents are implemented using the BDI4JADE platform¹², a BDI layer on top of JADE (Java Agent Development Framework) [nunes2011bdi4jade]
- We used JGAP (Java Genetic Algorithms Package) to implement our search-based approach.
- The example pairs are used for a UML2REL transformation problem.
- The main application is implemented in JAVA.

More concretely, *MoTrans-BDI* is composed of two parts, the backend which contains the BDI model and the frontend that is represented by the graphical user interface. The main component modules are listed below.

Architecture Technique

The system is deployed in a centralized architecture, on a Windows Server 2019 virtual machine. The BDI agents are implemented using the BDI4JADE platform, a BDI layer on top of JADE (Java Agent Development Framework).

BDI4JADE leverages all the features provided by JADE and reuses it as much as possible. Other highlights of our JADE extension, besides providing BDI abstractions and the reasoning cycle, include :

- Use of Capabilities – agents aggregate a set of capabilities, which are a collection of beliefs and plans, and allow modularisation of particular agent functionality.
- Plan Body is an Extension of JADE Behaviour – in order to better exploit JADE features, plan bodies are subclasses of JADE behaviours.
- Java Annotations – annotations are provided to allow easier configuration of agent components, without compromising its flexibility.

11. <https://anonymous.4open.science/r/MoTrans-BDI-728E>

12. <https://jade.tilab.com/>

- Extension Points – strategies can be easily implemented to extend parts of the reasoning cycle, such as belief revision and plan selection.
- Listeners and Events – different events (such as events related to goals and beliefs) can be observed in the platform, allowing listeners to react according to events that occurred.
- Java Generics for Beliefs – beliefs can store any kind of information and are associated with a name, and if the value of a belief is retrieved, it must be cast to its specific type, so the use of Java generics allows us to capture incorrect castings at compile time.

As opposed to different BDI platforms that have been proposed, it does not introduce a new programming language nor rely on a domain-specific language (DSL) written in terms of XML files. Because agents are implemented with the constructions of the underlying programming language, Java, we bring two main benefits, as detailed below.

- Features of the Java language, such as annotations and reflection, can be exploited for the development of complex applications.
 - It facilitates the integration of existing technologies, e.g. frameworks and libraries, which is essential for the development of large scale enterprise applications, involving multiple concerns such as persistence and transaction management. This also allows a smooth adoption of agent technology.
-

MoTrans-BDI - Call for proposal service

As depicted in Figure [Figure 7.1a](#), the user begins by selecting a transformation type (e.g. Petri-net2FSM, UML2REL), then he/she loads an instance of a SM which is stored as an XMI file with the possibility of editing the concepts of this instance according to its metamodel. To transform this model, the user launches the MA agent which in turn encapsulates this request as a triplet (SMM, SM, TMM) and launches the call for proposal which will wake up the BTAs.

MoTrans-BDI - SM-transformation service

Figure [Figure 7.1b](#) shows the MoTrans-BDI SM-transformation service. After receiving the CFP message from the MA agent, the BTA agents begin by filtering their own examples' base that represents the beliefs in order to produce a subset of examples that represents the desires of each agent. All selected pairs contain the SM given as input or a similar of it. Each BTA uses a genetic algorithm to find a set of good target models that minimize the number of conformance errors and maximize the number of satisfied heuristics. At the end of this process, the MoTrans-BDI user can export the BTA proposal as an XML file. Finally, each BTA's best TM is sent to the MA agent.

MoTrans-BDI - Refinement service

The panel showed in Figure [Figure 7.2](#) presents the refinement service provided by the MA agent. The MA agent uses an automatic analysis based on a XMI parser that extracts the TM elements required for the refinement in a form of a feature vector. Immediately after, the MA evaluates the feasibility

of each TM by checking existing heuristics to return the best target model (TM^{Best}) that satisfies the maximum of the heuristics.

MoTrans-BDI - Configuration service

In a multi-agent system where agents are distributed, some configuration is required to allow agents to migrate towards machines that contain example pairs. As shown by the configuration service panel of Figure [Figure 7.2](#) Ⓛ, the user creates first the requested number of BTA agents to include them in the CFP protocol, and then he decides where to migrate each BTA agent (in this example all BTA agents are located in the local host). The system also displays a log file to enable users to track ACL (Agent Communication Language) message exchanging between agents. Designers can then analyze the log journal generated by MoTrans-BDI, check the logic behind the SM transformation, and make the correct decision.

7.2.2 Evaluation of MoTrans-BDI

We conducted several experiments to evaluate the proposed approach against the final solution as well as the BTAs' intermediate TMs.

7.2.2.1 Creating the dataset

Due to the lack of good datasets $\mathcal{D}(SM, TM)$, we have been constrained to build a homemade dataset for the purpose of testing and evaluating our proposal. To create our dataset that represents the BTAs' beliefs, we asked researchers who are knowledgeable about the MDE paradigm to provide transformation examples. To this end, a same set of source models was sent to all participating researchers to complete them with the corresponding target models, which results in example pairs (SM, TM) . We also asked them to consider a set of heuristics, corresponding to different QoS attributes, expressed in a pre-established grid of heuristics related to each SM. Examples obtained from researchers have been encoded in XMI (XML Metadata Interchange) format. Other examples have been collected, from online documentation and discussion forums. Examples pairs were grouped according to their sources with each group considered as an expert view and stored in a unique repository to feed the beliefs of just one BTA. The heuristics were stored to be used for refining target models. We manually inspected the obtained data to check and correct any typos and modeling errors. Figure [Figure 7.3](#) presents the process of gathering data and the creation of the dataset $\mathcal{D}(SM, TM)$ that we used as BTAs' beliefs in our approach.

7.2.2.2 Evaluation's Data Collection

With the aim of checking if MoTrans-BDI emulates human's transformation process in a situation with a lack of transformation rules, we have prepared a model transformation test that contains a set of given SMs to be transformed. We have also integrated some questions in the test by asking the participants about the preferences on TM choosing. The process to get the experiments data collection followed four main steps :

1. *Participants : Invitation of experts in model transformation.* We selected three participants from academia for which model transformation is a main topic of interests (e.g., MODELS : *Model Driven Engineering Languages and Systems*). Additionally, the participants must have produced at least one scientific paper on model transformation in the past. The challenge to take up for participants was to transform the given SMs without using a prior transformation rules.
2. *Hints about the transformation's goal :* In order to guide the participants during the model transformation process, we gave them some hints/recommendations about the transformation regarding QoS attributes such as performance, security, reusability etc. For instance, a UML2REL model transformation with power hints to help participants proposing a TM that optimizes the queries' execution cost in terms of energy consumption.
3. *Model transformation collaboration :* Elaborating a collectively designed TM consists of two rounds : (1) Performing individual model transformation, and (2) Performing a shared design model transformation.
4. *Collecting Data :* After producing individual TMs and the collective final TM, experts' data was collected to be used in MoTrans-BDI testing.

7.2.2.3 Evaluation Metrics

The *MoTrans-BDI* is implemented using JADE Multi-agents platform, the genetic algorithm library, and EMF compare ¹³. We compared the target models produced by MoTrans-BDI to the participants' individual and collective target models. In order to calculate the evaluation metrics such as *F-Measure* (F_{trans}), *Accuracy* (ACC_{trans}), *Precision* (P_{trans}) and *Recall* (R_{trans}), we used two recommended parameters, namely, MoTrans-BDI's correct TM instances and incorrect TM instances compared to the experts' collectively designed TMs in terms of correct and incorrect application of model transformation operations (i.e. split, merge, create, and delete).

Especially, for each operation type that represents the change made on the *SM* (e.g. create, update, or delete), we count the number of the *TM*'s fragments that were correctly transformed (TP/TN) and incorrectly predicted (FP/FN). Formally, we define these four metrics' parameters as follows :

- **True Positive (TP)** : This parameter refers to the number of predictions where MoTrans-BDI uses an operation type Op_x (e.g. split) for the model transformation process, and model transformation experts recommend the same operation type (Op_x).
- **True Negative (TN)** : This parameter refers to the number of predictions where MoTrans-BDI uses an operation type different from Op_x ($\neg Op_x$), and model transformation experts recommend also an ($\neg Op_x$) operation.
- **False Positive (FP)** : Refers to the number of predictions where MoTrans-BDI uses an operation type different from Op_x ($\neg Op_x$), and model transformation experts recommend the operation type Op_x .
- **False Negative (FN)** : Refers to the number of predictions where MoTrans-BDI uses an operation type (Op_x) whereas model transformation experts recommend an operation type different from Op_x ($\neg Op_x$).

13. <https://www.eclipse.org/emf/compare/>

Finally, we can calculate the above mentioned metrics by considering all used operations.

$$ACC_{trans} = \frac{TP + TN}{TP + TN + FP + FN} \quad (7.1)$$

$$P_{trans} = \frac{TP}{TP + FP} \quad (7.2)$$

$$R_{trans} = \frac{TP}{TP + FN} \quad (7.3)$$

$$F1_{trans} = \frac{2 * P * R}{P + R} \quad (7.4)$$

7.2.2.4 Experiment strategy and procedure

This setup includes the different scenarios considered in our experiment procedure. The experiments we conducted aimed at evaluating the validity of the following MoTrans-BDI's main features : (1) *BTA Accuracy*, by comparing each BTA's produced TM with the experts' collectively designed TM (i.e. evaluation of the genetic algorithm based model transformation), (2) *MoTrans-BDI Accuracy* to show how much MoTrans-BDI's generated TM is close to the experts' collective TM, and (3) *Collective design Accuracy* to assess collective design and individual design compared to experts' design. Finally, we conducted an experience using the SSB Benchmark to evaluate the Motrans-BDI solution against the response time as a QoS attribute. [\[sanchez2016review\]](#).

7.2.2.5 BTAs Model Transformation Correctness Testing

We present a preliminary study of the results of our approach in terms of correctness. To simplify the presentation, we use UML2REL model transformation. The correctness of BTAs' TMs is studied through its accuracy compared to the experts' collectively designed TMs. While each BTA's TM is produced by applying genetic operators, namely, crossover, mutation, and selection, the experts' collectively designed TM is based on their experience and background in model transformation. The challenge to take up for the BTAs is to emulate human reasoning. The best way to illustrate the BTA's produced TM correctness is to consider different scenarios of SM availability in BTA's data store since genetic operators require an important initial population of example pairs (high rate of availability, average rate of availability, and low rate of availability). That is to say, the rate of availability is high if the percentage of TMs, in the initial population, having an SM equal to the SM to be transformed is upper than 60%. A rate of availability is average if the percentage is between 40% and 60% and it is considered as low if the percentage is lower than 40%. To this end, we selected 12 SMs with different rate of availability to be transformed by the BTAs and experts simultaneously. The results show that the average similarity is about 69,7%, 59,8%, 46%, for BTA1, BTA2 and BTA3 respectively. While the average deviation is quite important, it is about 41,5%. Indeed, the similarity between TMs varies according to the considered example pair rate of availability and the rate of heuristics satisfaction. Figure [Figure 7.4](#) shows the similarity between the BTAs' 12 produced TMs and the experts' collectively designed TMs. For instance, the similarity between the BTA-1's TM1 and experts' TM1 is very high (similarity = 85%) owing to the high number of example pairs (SM,TM) in the BTA-1's data store (32 example pairs), whereas for roughly

the same number of example pairs the similarity for BTA-2's TM1 is only high (similarity = 72%). This is explained by the fact that the major TM part of examples pairs do not satisfy completely the heuristics (35%). In conclusion, the more example pairs satisfying the heuristics we provide in the BTA's data store, the better our BTAs transform models.

Along with the BTAs Model Transformation Correctness Testing, we evaluated the time execution cost of the approach according to the three cases of SM availability. The reason behind this criteria of calculation is that the approach can be evaluated in the worst case scenario (Low rate of SM availability), the best case scenario (High rate of SM availability), and intermediate assumptions (Average rate of SM availability). Calculating the time execution cost of our approach amounts to estimating the time needed to propose a solution TM for each BTA. We have taken the same example of the twelve SMs, where each of them is annotated by its rate of availability into the corresponding BTA's data store ("H" for high rate of availability, "A" for average, and "L" for low), and we have calculated the BTA's execution time of each SM in minutes. The results shown in table Tableau 7.1 indicate that the time cost is low for SMs with high rate of availability (e.g. TM1(H,2,5), TM2(H,3,33), TM3(H,2,83)). This is due to the fact that the genetic algorithm converges quickly to the best TM. Whereas higher time costs are related to SMs with low rate of availability (e.g. TM1(L, 10,83), TM2(L, 10,83)). In this case the genetic algorithm achieves the best TM after crossing and mutation of several generations.

Intermediate time costs are obtained for most SMs with average rate of availability.

Note that MoTrans-BDI is dedicated for specific problems like co-design in companies, where experts adopt a shared design model transformation, that does not have a strict time constraint.

Example	TM1	TM2	TM3	TM4	TM5	TM6	TM7	TM8	TM9	TM10	TM11	TM12
BTA 1	(H,2,5)	(H,3,33)	(H,2,83)	(H,3,33)	(H,5,17)	(H,5,00)	(H,3,83)	(H,4,67)	(H,4,17)	(H,3,33)	(H,5,00)	(H,5,67)
BTA 2	(H, 4,67)	(H,5,00)	(H,5,00)	(A,8,33)	(H,6,67)	(H,6,17)	(H,6,67)	(H,6,33)	(H,5,67)	(H,6,00)	(H,5,00)	(H,5,33)
BTA 3	(L, 10,83)	(L,10,83)	(A,10,00)	(H,6,67)	(A,7,50)	(H,6,67)	(A,8,83)	(A,8,00)	(A,7,33)	(A,8,33)	(A,7,50)	(A,7,00)

TABLEAU 7.1 – MoTrans-BDI Response time costs

7.2.2.6 MoTrans-BDI Accuracy Testing

In order to test the whole model accuracy (MoTrans-BDI accuracy), we used a testing dataset \mathcal{D}_{Test} that contains the same 12 SMs used in the first experiment. Each of these SMs has a different model signature. We used MoTrans-BDI to suggest for each input instance SM_x the appropriate TM_y . All MoTrans-BDI's generated TMs were classified as correct TM instances, and incorrect TM instances. We used two scenarios for calculating the similarity degree between MoTrans's TMs and experts' TMs, namely, with refinement and without refinement. The refinement process is required when many agents suggest different solutions for the same type of fragment (e.g. *many to many*). This process is applied to achieve the best results.

Table Tableau 7.2 summarizes the results of accuracy, recall and F1 measure for the both model transformation scenarios. The with-Ref accuracy is 0.96 and the recall and F1 measure achieve to 0.75 and 0.71 respectively, while the Without-Ref accuracy attains 0.75 and the recall and F1 measure are estimated to 0.70 and 0.69 respectively. These results demonstrate that using heuristics is advantageous in model transformation and has an impact on enhancing the similarity between experts' TMs and MoTrans-BDI's TMs. The experts explained that they used these heuristics implicitly when asking to

transform models regarding some QoS attributes. Furthermore, from Figure Figure 7.5 we can observe that the similarity for TM1, TM2, TM3, and TM7 is very high whereas it is roughly high for the most remaining TMs. The fact that the TM part of example pairs satisfies the heuristics or not impacts directly on the selection of this TM as a candidate solution. The TM similarity showed in Figure Figure 7.5 could be different if we consider other heuristics according to different QoS attributes. Generally, MoTrans-BDI emulates well human experts in model transformation by example, however there are some exceptions where the similarity is low despite the fact that the TM satisfies the heuristics (see TM11). This case occurred because SM11 does not appear in any of the BTA's beliefs and the SMs similar to SM11 taken as a first population for the genetic algorithm have a low similarity.

Model	Precision	Recall	F-measure	Correctly/Incorrectly classified instances
Without-Ref	0.75	0.705	0.692	(82.88%, 7.12%)
With-Ref	0.96	0.755	0.713	(81.08%, 18.91%)

TABLEAU 7.2 – Summary of *MoTrans-BDI* comparison metric values

Additionally, we explored the different experiment's scenarios used by the MA to choose the best TM in order to identify the most dominant refinement scenario. We noted that while *Partial Transformation* ranks first with 55%, *With conflict* only ranks minute with 30% just before *Same decision* with 15%. We can provide three potential explanations for this situation. First, each BTA's proposal is based on a partial transformation of the SM and the TM recommended by the MA is a form of complementarity between the three BTA's proposals. This is due to distinct believes (SM,TM) of each BTA, or because examples of each BTA are intended to a particular transformation type. Second, the *With conflict* scenario appear when the BTAs' decisions are contradictory. This is owing to the different experts' design view that reflects distinct heuristics behind each proposal. Third, the situation of the same decision is obtained when the input SM is a common and recognized example. In other words, the problem of transformation type is mature enough due the consensus between a multiple point-of-view.

7.2.2.7 Collective design Vs Individual design

The aim of the last experiment is to assess collective design (MA's TM) and individual design (BTA's TMs) compared to experts' design. In this experiment we used results from experiment 1 and experiment 2, we aggregated the similarity of all TMs in a representative average similarity for each agent. The results shown in Figure Figure 7.6 indicate that the MA's produced TMs are generally closer to the experts transformation design than the BTA's proposed TMs. This result strengthened the whole model philosophy that promotes the multi-view design rather than individual design.

7.2.3 Evaluation of MoTrans-BDI proposal's impact against Performance Measurement

In this experiment, we present a case study of MoTrans-BDI to recommend a target model that satisfies the response time as QoS attribute. This means that we shall discuss the applied heuristics validity against the response time QoS attribute. We consider a scenario of the source model presented

in the top Figure ?? of the motivating example (cf. Section ??). The choice of this source model is motivated by the Conventional Database Schema, namely, the SSB benchmark [sanchez2016review] and the possibility to generate sample data randomly with different scales (SSB scale factor, ranging from 1 GO to 10000+ GO) using the available tool QueryGen of SSB, which we use to generate data and workload for each TM proposed by MoTrans-BDI. The SSB schema used in our experience contains five classes, namely, *Lineorder*, *Customer*, *Supplier*, *Part*, and *Dates*. The considered QoS performance attribute is represented by several heuristics to be satisfied such as Merging/Splitting the SM's classes regarding a defined threshold number of relations or reducing the number of joins required for insert, update, and delete statements.

After retrieving the recommended TMs from the BTAs and the MA, we materialize each TM in a DBMS (i.e. oracle DBMS). We empirically evaluate the set of candidate solutions to compare their response times against the reference TM's response time (i.e. called the initial TM) using a sample of twelve queries against the schema. Figure Figure 7.7 shows the initial TM that corresponds to the SSB's SM. In this case, each class of the SSB's SM is mapped into its corresponding relation. We consider this solution as a reference to see if each MoTrans-BDI's proposal increases or decreases the performance cost of the SSB's workload.

Commonly, each TM proposed by the BTAs and the MA is generated by split-merge of the surrounded attributes of each relation in the initial TM (see Figure Figure 7.7). For instance, when we transform the attributes surrounded by a red box (e.g. attributes : city, nation, and region of the relation *Customer*) into a new relation, this can be benefit or nor for a given query depending on its selectivity

For each agent's proposal, we rewrite the original query workload according to the proposed TM solution. For example, Listing Liste 7.1 shows an example of a query that corresponds to the initial TM presented in Figure Figure 7.7 and Listing Liste 7.2 shows the Listing Liste 7.1's query rewrite according to a TM proposed by an agent.

```

1 SELECT SUM( LO_EXTENDEDPRICE* LO_DISCOUNT ) AS revenue
2 FROM DIM_DATES , FACT_LINEORDER
3 WHERE H_D_YEAR = 1993 AND LO_DISCOUNT >= 1 AND LO_DISCOUNT <= 3 AND LO_QUANTITY < 25 AND LO_ORDERDATE = D_DATEKEY

```

Listing 7.1 – Excerpt of the SQL query corresponding to intial model presented in Figure Figure 7.7

```

4 SELECT SUM( LO_EXTENDEDPRICE* LO_DISCOUNT ) AS revenue
5 FROM FACT_LINEORDER, D_YEARDim3, D_MONTHDimYEAR4, DIM_DATES-MONTH
6 WHERE H_D_YEAR = 1993 AND LO_DISCOUNT >= 1 AND LO_DISCOUNT <= 3 AND LO_QUANTITY < 25 AND LO_ORDERDATE = D_DATEKEY
    AND H_D_YEARID = H_D_YEARID_FK AND H_D_MONTHID = H_D_MONTHID_FK

```

Listing 7.2 – Excerpt of rewrite of the SQL query of Listing Liste 7.1 according to a new TM

In order to compare the quality of TMs provided by BTAs and the MA with that of the initial TM, we ran the twelve queries of the SSB benchmark with a scale factor of 10 GB. The response time of each query is shown in Table Tableau 7.3. In addition, Figure Figure 7.8 shows the aggregation of the twelve queries workload response times, in milliseconds, corresponding respectively to the initial TM, the three BTAs, and the MA. We can see that the best performance cost is that one corresponding to the MA. Although, the BTAs' performance cost also outclasses the initial TM's one. This difference perceived on the performance cost shows that using MoTrans-BDI still makes sense in optimizing the query performance.

As mentioned above, in order to evaluate the impact of the heuristics on the query workload

performance measurement, we analyzed the response time of the original queries and the rewrited queries. As we can easily see in Table [Tableau 7.3](#), the MA proposal leads, in most cases, to lower query cost compared to those achieved by queries related to the BTAs' proposals and the initial TM. By analyzing the signature of each query, we remark that the BTAs' proposals have a positive impact on queries that are more selective (Q2,Q3,Q4,Q5,Q6,Q7,Q8,Q11), that is, the heuristics applied for splitting the initial TM to optimize the query join operations are benefit. In contrast, BTAs' proposals are not beneficial for not selective queries (Q1,Q9,Q10,Q12) which promote the data grouping in the same relation. Consequently, the heuristics considered by BTAs, which are not reducing the number of joins, have a negative impact on these type of queries. In summary, the integration of solutions with a refinement process may provide a quality-driven model transformations based on domain assumptions and heuristics considered by MT designers.

Query	Initial TM	BTA1	BTA2	BTA3	MA
Q1	1357	1377	1377	1398	1277
Q2	170	185	185	175	163
Q3	102	110	110	98	98
Q4	703	675	675	740	672
Q5	650	638	638	631	620
Q6	345	330	330	350	305
Q7	310	295	295	270	270
Q8	550	544	544	539	490
Q9	480	480	480	480	480
Q10	120	150	150	96	96
Q11	910	803	773	773	773
Q12	420	360	360	402	360

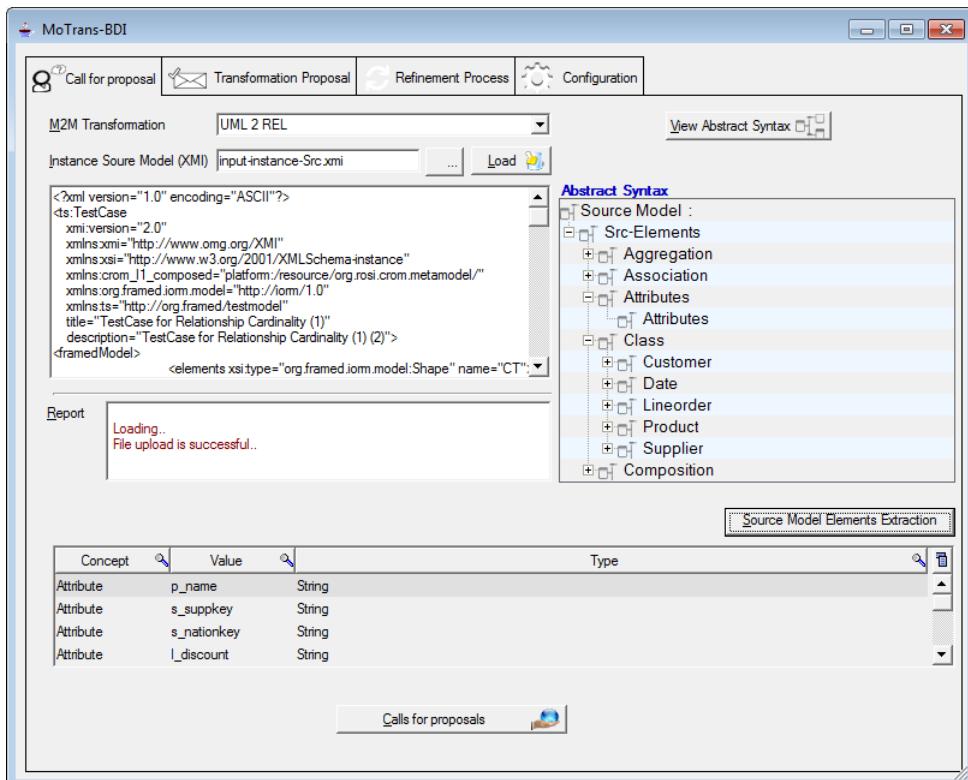
TABLEAU 7.3 – SSB benchmark queries response time estimations.

7.3 Threats to validity

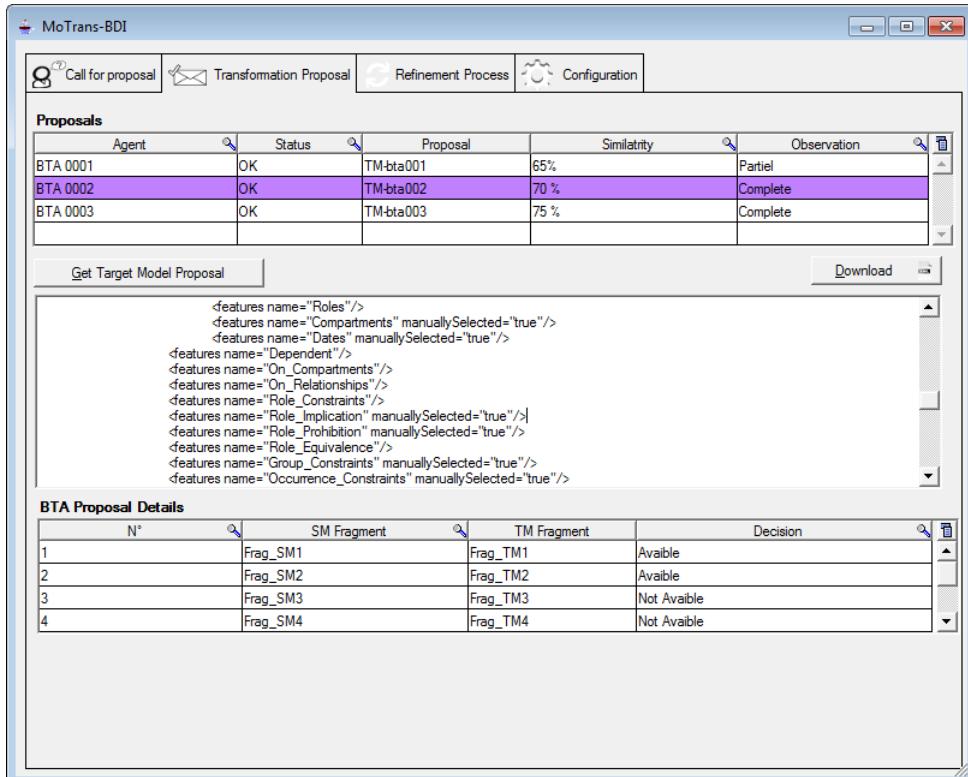
Some factors may threat the validity of our study. First we might have missed some examples due to the unavailability of a well known and exhaustive data-set of source and target model pairs related to a particular business DSL. To mitigate this threat, we need an open share repository to encourage scientists and practitioners from the academic and industrial area to provide and share examples generated by human expertise. Second, the method used for coding a TM solution must be adapted to each DSL manually which can be error task prone and time consuming. To mitigate this risk, we may use a generative and automatic interface as a regular transformation language that may allow the TM's coding (i.e. the CDS) to be tailored to a particular application context of a given DSL. The third potential threat to validity of our study is related to the impact of the evolutionary algorithm. In the genetic process, we might achieve the best results by applying the genetic primitives. In some cases, the solution may not converge to the best solution. To avoid a divergence of the proposed algorithm, the mutation and the crossover operators applied by BTAs must be driven by the transformation goal to ensure the conformance of the generated instances and to satisfy more of heuristics. In addition, MoTrans-BDI

can be extended by using other evolutionary algorithms to reproduce the TM instances such as Particle Swarm Optimization algorithm, Bee Colony algorithm, etc. These algorithms can be applied by the BTAs collectively or individually such that each strategy can be compared to others in terms of precision and accuracy. Finally, the last threat to validity concerns the time cost of MoTrans-BDI. This has implications for the execution efficiency of the approach, because the search-based derivation of the mapping needs to be re-executed for each different source model. However, this is time-consuming and inefficient in the case of a large number of possible tests. Indeed, to ensure the scalability of the comparison tests, it is preferable to reduce the comparison complexity to be polynomial or pseudo-polynomial at worst. So, to reduce the number of tests and to guarantee a good scalability of the analysis dataset, we suggest that the analysis dataset to be sorted by test family instances. This latter, allows to test only instances sharing the same model transformation problem (e.g. UML to REL, UML to Petri Net, Finite State Machine to Petri nets). Other techniques must be explored like caching a solution to avoid re-executing the mapping for each different source model.

Chapitre 7. Outil Support de notre cadre de transformation MTBE



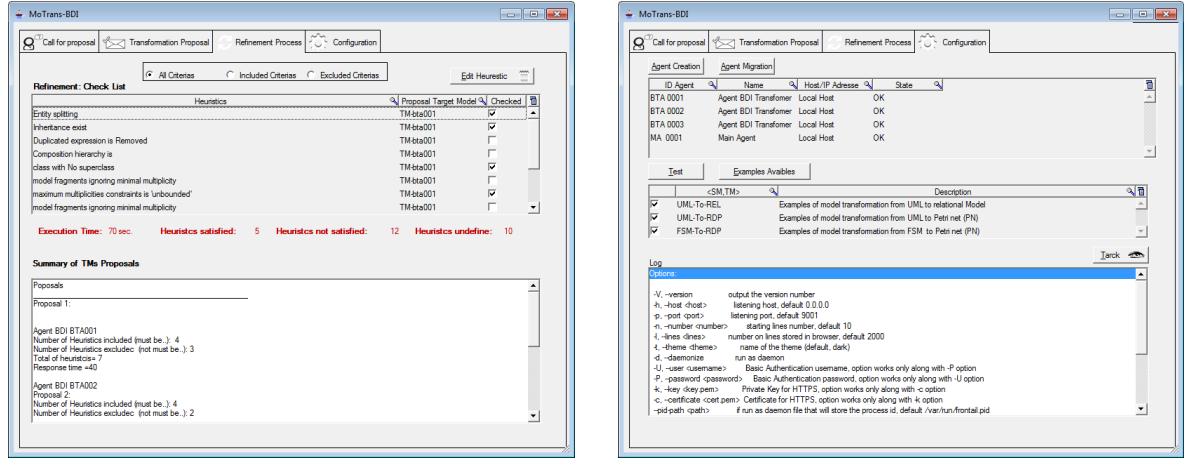
(a) MoTrans-BDI Call for proposal service



(b) MoTrans-BDI SM-transformation service

FIGURE 7.1 – MoTrans-BDI main GUI for CFP and Transformation services.

7.3. Threats to validity



(a) MoTrans-BDI Refinement service

(b) MoTrans-BDI Configuration service

FIGURE 7.2 – MoTrans-BDI main GUI for Refinement and Configuration services.

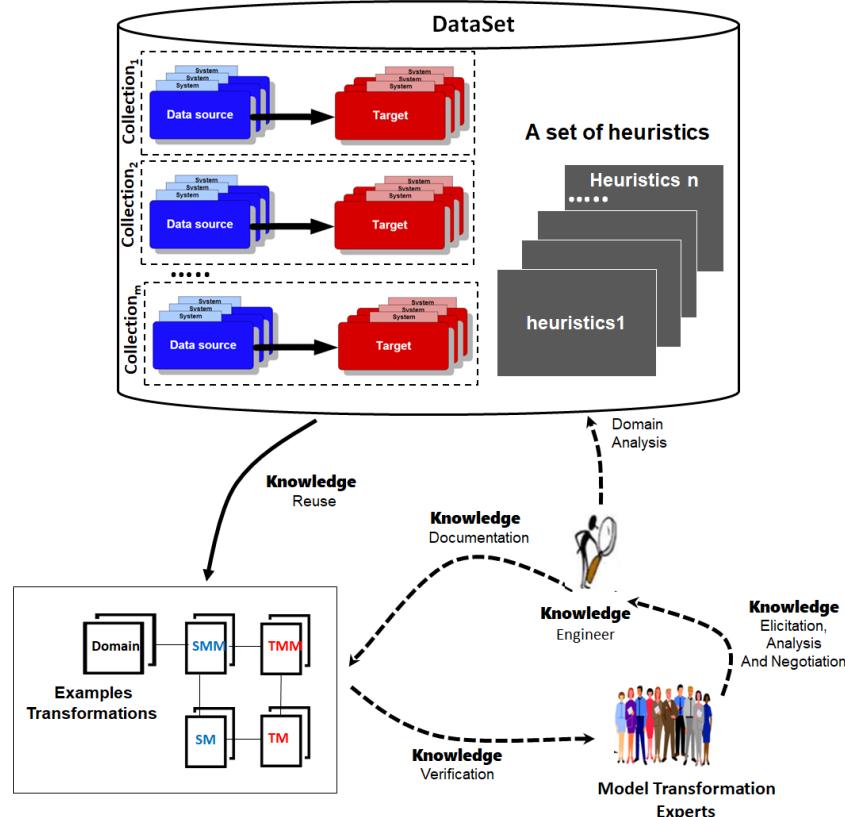


FIGURE 7.3 – Model Transformation Data gathering.

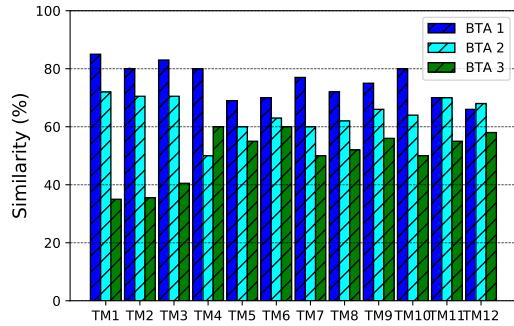


FIGURE 7.4 – BTAs Model Transformation Correctness Testing.

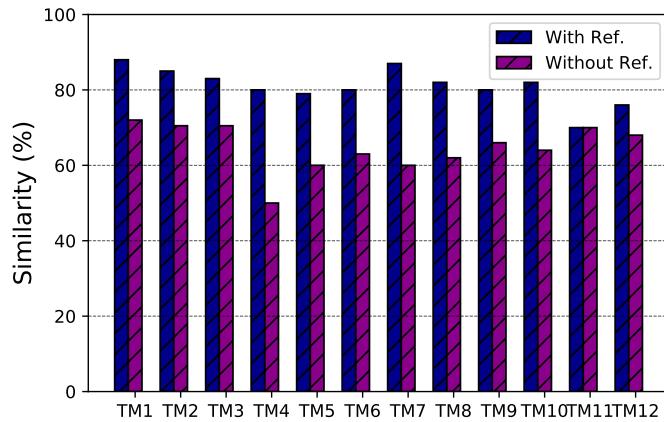


FIGURE 7.5 – TMs Similarity Degree with and without refinement.

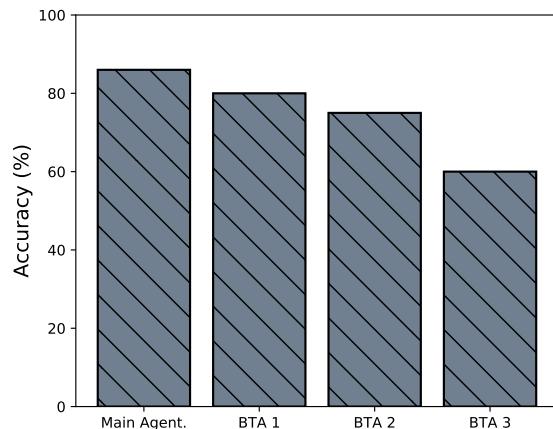


FIGURE 7.6 – MA's TMs similarity Vs BTAs' TMs similarity

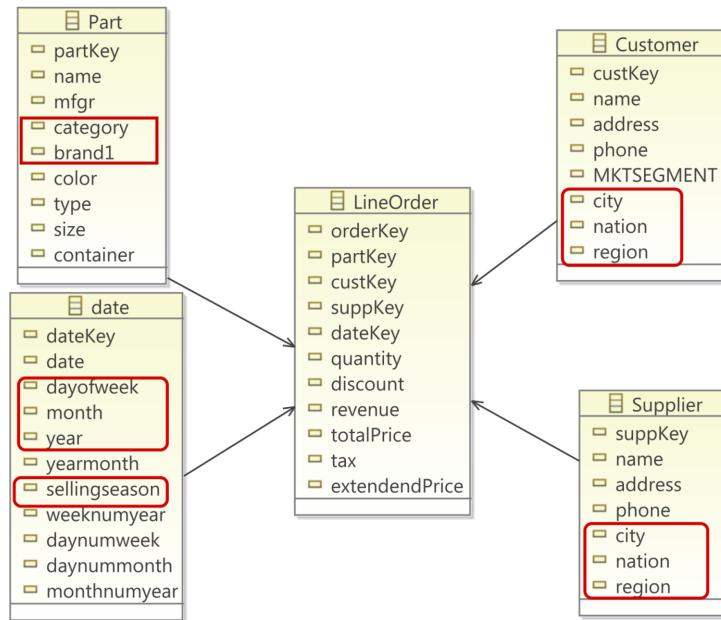


FIGURE 7.7 – The initial TM corresponding to the SSB’s SM

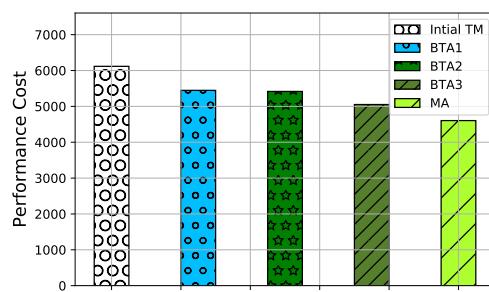


FIGURE 7.8 – Cumulative queries performance cost corresponding to the initial model, the BTAs, and the MA



Conclusion et Perspectives

« *The true method of knowledge is experiment.* »
— William Blake(1757-1827)

Sommaire

8.1	Introduction	84
8.2	Conclusion	84
8.3	Perspectives	84

8.1 Introduction

8.2 Conclusion

The main goal of this work was to design a framework called *MoTrans-BDI* for MTBE based on *Evolutionary Multi-Agent Systems*. Providing a collaborative process for this type of model transformation is challenging and crucial. Therefore, embedding BDI agents in a Contract Net Protocol system architecture helps us to create a combined final decision from individual and different design views. The BTA's beliefs are based on experts' knowledge in the domain of MTBE and are structured as example pairs (SM, TM). Although, in our experimentation all BDI agents use the same method to produce the TM, the idea of producing the final TM from experts with different strategies deserves to be deeply investigated. A first step of the approach consists of sending all needed data to BTAs in order to produce their own solutions. In the second step, the MA retrieves all BTAs' propositions and uses heuristics related to a recommended set of QoS attributes to generate the best TM corresponding to the initial SM. Another feature of our model is that it is designed in a generic way such that its modular architecture allows replacing easily some components without affecting the overall functionality, such as (1) the type of BTA's beliefs according to a specific MTBE, (2) the BTAs used algorithm that produces the TM, and (3) the plug-gable component in the MA model that embeds heuristics to refine the final TM.

We evaluated our model on twelve UML2REL transformation problems. We first compared the BTAs' produced TMs to the experts' collectively designed TMs. Then, we evaluated the MA's generated TMs compared to the experts' TMs, and finally we conducted a comparison between individually designed TMs and collectively designed TMs against the experts transformation design. Our validation shows that MoTrans-BDI generally produces good transformations on examples that are well represented in agents' beliefs and that highly satisfy heuristics. Despite these encouraging results, there is still room for improvement. Indeed, even if the approach minimizes the number of conformance errors in the target model, it can still yield non-conformant models. Finally, we empirically evaluated MoTrans-BDI's proposals on SSB benchmark to assess the heuristics impact on the response time as QoS attribute. Results show that MoTrans-BDI proposals reduce qualitatively corresponding queries response times than queries related to the reference model.

8.3 Perspectives

This work opens several directions of further research. From the theoretical point of view, we plan to incorporate a negotiation to resolve conflicts occurring during the transformation process, in particular with a large number of concurrent BDI transformer agents. From the methodological point of view, we are also planning to investigate the possibility to keep the designer in the loop of the transformation process and to allow him to browse the existing target models, to edit, to enrich new ones and to validate them. From the genericity perspective, we will explore the idea of using different MTBE algorithms for each BTA to enhance the quality of BTAS' produced TMs.

Quatrième partie

Conclusion et perspectives



Conclusion générale et Perspectives

« To accomplish great things, we must not only act, but also dream;
not only plan, but also believe. »
— Anatole France (1844-1924)

Sommaire

9.1 Conclusion	88
9.1.1 État de l'art	88
9.1.2 Contribution 1	88
9.1.3 Contribution 2	88
9.1.4 Preuve de concept	88
9.2 Perspectives	88
9.2.1 Ajout des nouvelles fonctionnalités	88
9.2.2 Qualité de MdCs partagés	88
9.2.3 Explorer l'analyse des relations entre les MdCs	88
9.2.4 Rapide prototype des MdCs	88
9.2.5 Utilisation de notre framework en mode pédagogique d'enseignement .	88

9.1 Conclusion

Nous présentons dans ce chapitre un bilan du travail que nous avons effectué ainsi qu'un ensemble d'ouvertures et de perspectives pour ce travail.

9.1.1 État de l'art

9.1.2 Contribution 1

9.1.3 Contribution 2

9.1.4 Preuve de concept

9.2 Perspectives

Les travaux initiés dans cette thèse peuvent se poursuivre dans de nombreuses directions. Nous esquissons ici quelques pistes de perspectives à court et à long terme.

9.2.1 Ajout des nouvelles fonctionnalités

9.2.2 Qualité de MdCs partagés

9.2.3 Explorer l'analyse des relations entre les MdCs

9.2.4 Rapide prototype des MdCs

9.2.5 Utilisation de notre framework en mode pédagogique d'enseignement

Cinquième partie

Annexes

**Questionnaire Académique en ligne de
l'évaluation de l'acceptabilité de MetricStore**



A.1 Questionnaire Académique

ANNEXE

B

Lexique



Lexique





Bibliographie

- [1] I. BAKI et H. SAHRAOUI. « Multi-step learning and adaptive search for learning complex model transformations from examples ». In : *ACM Transactions on Software Engineering and Methodology (TOSEM)* 25.3 (2016), p. 1-37 (cf. p. 4, 36).
- [2] I. BAKI et al. « Learning implicit and explicit control in model transformations by example ». In : *International Conference on Model Driven Engineering Languages and Systems*. Springer. 2014, p. 636-652 (cf. p. 34, 37).
- [3] Z. BALOGH et D. VARRÓ. « Model transformation by example using inductive logic programming ». In : *Software & Systems Modeling* 8.3 (2009), p. 347-364 (cf. p. 4, 32).
- [4] G. BERGMANN et al. « Incremental pattern matching in the VIATRA model transformation system ». In : *Proceedings of the third international workshop on Graph and model transformations*. 2008, p. 25-32 (cf. p. 4).
- [5] K. BERRAMLA, D. BENHAMAMOUCH et al. « Model transformation generation a survey of the state-of-the-art ». In : *2016 International Conference on Information Technology for Organizations Development (IT4OD)*. IEEE. 2016, p. 1-6 (cf. p. 4).
- [6] J. BÉZINVIN et OTHERS. « Model transformations ? transformation models ! » In : *International Conference on Model Driven Engineering Languages and Systems*. Springer. 2006, p. 440-453 (cf. p. 4).
- [7] M. BIEHL. « Literature study on model transformations ». In : *Royal Institute of Technology, Tech. Rep. ISRN/KTH/MMK 291* (2010) (cf. p. 4).
- [8] E. BOUSSE et OTHERS. « Execution framework of the GEMOC studio (tool demo) ». In : *Proceedings of the 2016 ACM SIGPLAN International Conference on Software Language Engineering*. 2016, p. 84-89 (cf. p. 4).
- [9] A. A. BRAHIM, R. T. FERHAT et G. ZURFLUH. « Approche dirigée par les modèles pour l'extraction automatique du modèle NoSQL ». In : *Business Intelligence & Big Data : 15ème Edition de la conférence EDA, Montpellier France 2019*. BoD-Books on Demand. 2020 (cf. p. 10).
- [10] M. BRATMAN. « Intention, belief, and instrumental rationality ». In : *Reasons for action* (2009), p. 13-36 (cf. p. 42).
- [11] M. BRATMAN. « Intention, plans, and practical reason ». In : (1987) (cf. p. 43).
- [12] A. BYRSKI et al. « Evolutionary multi-agent systems ». In : *The Knowledge Engineering Review* 30.2 (2015), p. 171-186 (cf. p. 5).
- [13] G. L. CASALARO et al. « Model-driven engineering for mobile robotic systems : a systematic mapping study ». In : *Software and Systems Modeling* (2021), p. 1-31 (cf. p. 4).
- [14] B. CHAIB-DRAA, I. JARRAS et B. MOULIN. « Systèmes multi-agents : principes généraux et applications ». In : *Edition Hermès* 242 (2001), p. 1030-1044 (cf. p. 5).
- [15] X. DOLQUES, M. HUCHARD et C. NEBUT. « Génération de transformation de modèles par application de l'ARC sur des exemples ». In : *LMO : Langages et Modèles à Objets*. Cépaduès Editions. 2009, p. 61-75 (cf. p. 33).

BIBLIOGRAPHIE

- [16] X. DOLQUES et al. « Learning transformation rules from transformation examples : An approach based on relational concept analysis ». In : *2010 14th IEEE International Enterprise Distributed Object Computing Conference Workshops*. IEEE. 2010, p. 27-32 (cf. p. 33).
- [17] M. EYSHOLDT et H. BEHRENS. « Xtext : implement your language faster than the quick and dirty way ». In : *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*. 2010, p. 307-309 (cf. p. 4).
- [18] M. FAUNES, H. SAHRAOUI et M. BOUKADOU. « Generating model transformation rules from examples using an evolutionary algorithm ». In : *2012 Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*. IEEE. 2012, p. 250-253 (cf. p. 34).
- [19] M. FAUNES, H. SAHRAOUI et M. BOUKADOU. « Genetic-programming approach to learn model transformation rules from examples ». In : *International Conference on Theory and Practice of Model Transformations*. Springer. 2013, p. 17-32 (cf. p. 34, 37).
- [20] I. GARCÍA-MAGARIÑO et al. « A tool for generating model transformations by-example in multi-agent systems ». In : *7th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS 2009)*. Springer. 2009, p. 70-79 (cf. p. 29, 36).
- [21] D. C. HALBERT. « Programming by example ». Thèse de doct. University of California, Berkeley, 1984 (cf. p. 28).
- [22] E. INSFRAN, J. GONZALEZ-HUERTA et S. ABRAHÃO. « Design guidelines for the development of quality-driven model transformations ». In : *International Conference on Model Driven Engineering Languages and Systems*. Springer. 2010, p. 288-302 (cf. p. 5).
- [23] F. JOUAULT et al. « ATL : A model transformation tool ». In : *Science of computer programming* 72.1-2 (2008), p. 31-39 (cf. p. 4).
- [24] G. KAPPEL et al. « Model transformation by-example : a survey of the first wave ». In : *Conceptual modelling and its theoretical foundations*. Springer, 2012, p. 197-215 (cf. p. 4).
- [25] A. KAVIMANDAN et A. GOKHALE. « Applying model transformations to optimizing real-time QoS configurations in DRE systems ». In : *International Conference on the Quality of Software Architectures*. Springer. 2009, p. 18-35 (cf. p. 4).
- [26] A. KAVIMANDAN et A. GOKHALE. « Automated middleware QoS configuration techniques using model transformations ». In : *2007 Eleventh International IEEE EDOC Conference Workshop*. IEEE. 2007, p. 20-27 (cf. p. 4).
- [27] S. KENT. « Model driven engineering ». In : *International conference on integrated formal methods*. Springer. 2002, p. 286-298 (cf. p. 10).
- [28] M. KESSENTINI et al. « Search-based model transformation by example ». In : *Software & Systems Modeling* 11.2 (2012), p. 209-226 (cf. p. 4).
- [29] A. LIKAS, N. VLASSIS et J. J. VERBEEK. « The global k-means clustering algorithm ». In : *Pattern recognition* 36.2 (2003), p. 451-461 (cf. p. 36).
- [30] S. LOLIGNIER et al. « Le canal Nav1. 9-Protéine clé pour la perception du froid et cible thérapeutique potentielle contre la douleur ». In : *médecine/sciences* 32.2 (2016), p. 162-165 (cf. p. 10).
- [31] J. PAVÓN et J. GÓMEZ-SANZ. « Agent oriented software engineering with INGENIAS ». In : *International Central and Eastern European Conference on Multi-Agent Systems*. Springer. 2003, p. 394-403 (cf. p. 36).
- [32] A. S. RAO, M. P. GEORGEFF et al. « BDI agents : From theory to practice. » In : *Icmas*. T. 95. 1995, p. 312-319 (cf. p. 42).
- [33] P. F. RUSSELL, T. R. RAO et al. « On habitat and association of species of anopheline larvae in south-eastern Madras. » In : *Journal of the Malaria Institute of India* 3.1 (1940) (cf. p. 48).

-
- [34] H. SAADA et al. « Generation of operational transformation rules from examples of model transformations ». In : *International Conference on Model Driven Engineering Languages and Systems*. Springer. 2012, p. 546-561 (cf. p. 4, 33).
 - [35] F. L. SIQUEIRA et P. S. M. SILVA. « Applying MTBE Manually : a Method and an Example. » In : *MDEBE@ MODELS*. 2013, p. 2-11 (cf. p. 4).
 - [36] M. STEEN, M. MANSCHOT et N. DE KONING. « Benefits of co-design in service design projects ». In : *International Journal of Design* 5.2 (2011) (cf. p. 4).
 - [37] E. SYRIANI, R. BILL et M. WIMMER. « Domain-Specific Model Distance Measures. » In : *J. Object Technol.* 18.3 (2019), p. 3-1 (cf. p. 48).
 - [38] G. TAENTZER. « AGG : A graph transformation environment for modeling and validation of software ». In : *International Workshop on Applications of Graph Transformations with Industrial Relevance*. Springer. 2003, p. 446-453 (cf. p. 4).
 - [39] H. V. TEGUIAK. « Construction d'ontologies à partir de textes : une approche basée sur les transformations de modèles ». Thèse de doct. ISAE-ENSMA Ecole Nationale Supérieure de Mécanique et d'Aérotechique-Poitiers, 2012 (cf. p. 10, 13).
 - [40] D. VARRÓ. « Modèle de transformation par exemple ». In : *Conférence internationale sur les langages et systèmes d'ingénierie pilotés par les modèles*, p. 410-424 (cf. p. 32, 33).
 - [41] M. ZADAHMAD et al. « Domain-specific model differencing in visual concrete syntax ». In : *Proceedings of the 12th ACM SIGPLAN International Conference on Software Language Engineering*. 2019, p. 100-112 (cf. p. 48).
 - [42] M. M. ZLOOF. « Query by example ». In : *Proceedings of the May 19-22, 1975, national computer conference and exposition*. 1975, p. 431-438 (cf. p. 28).