La continuité numérique basée sur l'apprentissage de modèles de transformation : Une approche d'interopérabilité dirigée par les modèles

Quentin Brilhault

HESAM UNIVERSITÉ

Arts et Métiers Sciences et Technologies

ECOLE DOCTORALE SCIENCES ET METIERS DE L'INGENIEUR
Laboratoire d'Ingénierie des Systèmes Physiques et Numériques
(LISPEN) - Campus d'Aix-en-Provence

THÈSE

présentée par : **Quentin BRILHAULT**

soutenue le : **06 Décembre 2023**

pour obtenir le grade de : **Docteur d'HESAM Université**

préparée à : **École Nationale Supérieure d'Arts et Métiers**

Spécialité : **Génie industriel**

La continuité numérique basée sur l'apprentissage de modèles de transformation : Une approche d'interopérabilité dirigée par les modèles

THÈSE dirigée par :
Monsieur ROUCOULES Lionel

et co-encadrée par :
Madame YAHIA Esma

Jury

M. Vincent CHEUTET	Professeur des Universités, DISP-lab, INSA de Lyon	Président
M. Hervé PANETTO	Professeur des Universités, Université de Lorraine	Rapporteur
M. Julien LE DUIGOU	Professeur des Universités, Roberval, UTC	Rapporteur
Mme. Esma YAHIA	Maître de Conférences, LISPEN, ENSAM Aix	Examinatrice
M. Lionel ROUCOULES	Professeur des Universités, LISPEN, ENSAM Aix	Examinateur
Mme. Françoise PERREL	Chargée de mission, CNES Toulouse	Examinatrice
M. Raphaël BOISSONNADE	Ingénieur R&D, Thales Alenia Space Cannes	Examinateur

THÈSE

À force de croire en ses rêves, l'homme en fait une réalité. Hergé, 1969

Remerciements

Tout d'abord, je tiens à remercier le Centre National d'Etudes Spatiales et Thales Alenia Space de m'avoir permis de démarrer puis d'achever cette formidable aventure. Aventure que je n'aurais jamais pensée atteignable plus jeune.

Je remercie chaleureusement les membres de mon jury de thèse, le Professeur PANETTO Hervé, le Professeur CHEUTET Vincent, et le Professeur LE DUIGOU Julien, pour leur temps, leur attention et leurs précieux conseils. Leurs évaluations et leurs suggestions ont contribué à renforcer la qualité de ce travail.

Mes premiers mots seront naturellement adressés à Monsieur Raphaël BOISSONNADE sans qui cette aventure n'aurait jamais débutée. Il a placé sa confiance en moi, a été le premier à croire en notre projet, et s'est battu pour faire valoir nos idées. La période d'avant thèse, pendant laquelle nous construisions le sujet de recherche, traduit bien l'idée qui se cache derrière le dicton "*On a rien sans rien !*", où, malgré les embûches, il faut continuer d'avancer et d'y croire. Une bonne dose de persévérance, un peu de culot (parfois), de l'audace et surtout beaucoup de passion, c'est le cocktail parfait qui nous a permis de lancer cette thèse sur les bons rails. Alors merci Raphaël d'y avoir cru tout autant que moi.

Je ne saurais oublier Madame Françoise PERREL, qui m'a permis d'aspirer à des horizons encore plus vastes en m'ouvrant les portes du CNES. Dans le passé, mes rêves étaient alimentés par les reportages du CNES diffusés à la télé. Aujourd'hui, j'ai eu l'opportunité de collaborer avec les membres de cette institution historique, de m'impliquer dans sa vie interne, et de rencontrer de jeunes chercheurs comme moi tout aussi fascinés par l'espace et ses merveilles. Je te remercie Françoise pour l'attention que tu as portée à mes travaux, et surtout, pour la bienveillance et le soutient humain que tu m'as accordés.

J'en viens à présent à Madame Esma YAHIA, qui m'a été d'une aide et d'un soutien précieux. Sa porte était constamment ouverte pour offrir son assistance, et ce, même lorsque son emploi du temps était déjà surchargé. Je te remercie de m'avoir soutenu dans mes initiatives et de m'avoir poussé à approfondir mes idées. Ce manuscrit de thèse ne serait pas aussi vivant et pertinent sans ton expertise.

L'engagement indéfectible d'Esma, de Françoise et de Raphaël tout au long de ces trois années a permis la réussite et la rigueur qui découlent de ce manuscrit de thèse.

Je remercie également toutes les personnes qui ont participées de près ou de loin à cette thèse : Frédéric MONCLAR pour ses encouragements et son sens du partage, Nicolas SIVERA et Charley

SILLORAY pour m'avoir apporté leur aide tout au long de la création des différents démonstrateurs, et Julien REY pour son investissement, ses conseils et sa générosité. Sans Julien, je n'aurais probablement jamais eu l'opportunité de présenter mes travaux lors d'un groupe de travail de l'Agence Spatiale Européen. Je ne vais pas vous mentir... c'était le moment le plus stressant de la thèse, mais l'expérience en valait la chandelle. Merci Julien d'avoir donné une tout autre envergure à mes recherches.

Une thèse c'est comme embarquer dans un navire pour une expédition. Le trajet est rythmé par les tempêtes et les accalmies. Nous, les doctorants, sommes tous dans le même bateau, alors je remercie particulièrement les doctorants du laboratoire LISPEN qui ont navigués à mes côtés pendant trois ans, qui ont écoutés mes problèmes et mes réussites, et avec qui j'ai pu partager des discussions stimulantes qui ont contribuées à l'enrichissement de ce travail. Je souhaite également exprimer ma gratitude envers les membres du laboratoire LISPEN qui m'ont accueilli dans leur équipe, qui m'ont partagé leurs idées, et transmis leurs compétences.

Enfin, je suis profondément reconnaissant d'avoir eu la chance de travailler avec Monsieur Lionel ROUCOULES, mon directeur de thèse. Lionel m'a permis de bénéficier de son savoir et de sa vision de la recherche académique. Son influence positive sur l'ensemble des mes travaux est visible sur chacune des pages de ce manuscrit. Je te remercie sincèrement de m'avoir permis de réaliser pleinement mes idées. Ton approche pédagogique et ton engagement pour la recherche m'ont profondément influencé et m'ont donné envie, à mon tour, de poursuivre sur ce chemin. Tes enseignements resteront avec moi tout au long de ma carrière.

Je n'oublie pas ma famille et mes amis qui ont été d'un soutien constant tout au long de ce parcours. Leurs encouragements et leurs compréhensions ont été inestimables. Quelques dédicaces sont de rigueurs..

Commençons par les amis : Sophie, Carla, Chloé, Axel et Alban, je vous remercie de m'avoir accueilli il y a trois ans au sein de votre groupe de jeunes femmes médecins dynamiques et hommes épicuriens vivants aux crochets de leurs compagnes. Je suis reconnaissant de vous avoir rencontrés. On part quand en Écosse ? Sophie, tu pourras mettre une musique de *k-pop* pour fêter ma soutenance de thèse... promis (mais une seule !), Carla je pense que la pause sur l'air d'autoroute dans les gorges du Verdon restera dans nos mémoires, Chloé il est quand notre premier triathlon ?, Axel (alias tonton) il tape toujours autant ce rosé, et Alban (alias SCH) je te suivrai partout pour cueillir des champignons.

Olivier et Robin, de prêt comme de loin, je sais que vous serez indéfiniment à mes côtés. Vous êtes les exemples parfaits des amis qui deviennent la famille.

En parlant de famille, je remercie mes beaux parents, Christine et Gilles, ma belle soeur Florence et mon beau frère Thomas, sans oublier mamie Jacky et papy Charles, d'avoir suivi avec intérêt mes péripéties sans forcement tout comprendre. Je vous rassure, c'est normal... moi même j'ai du mal avec certains points. Plus particulièrement, je vous remercie de m'avoir si bien intégré au sein de la famille Baccino. Je suis fière d'être l'un des vôtres.

Je remercie mes parents, qui m'ont dit que tout était toujours possible à condition de fournir les efforts nécessaires pour y parvenir. Je repense souvent aux mauvaises notes que j'ai eues en orthographe à l'école, ainsi qu'aux soirées entières passées à étudier les tables de multiplication ou à apprendre mes

leçons d'histoire et de géographie. Le chemin fut long pour se construire. Je peux affirmer que je n'y serais jamais parvenu sans toi papa et sans toi maman. Sans oublier mes frères Thibault et Bastien aux yeux de qui je resterai sûrement un extraterrestre (ou le futur président), mes belles soeurs, Mélanie et Margaux et enfin, la petite Amaya, 1 an déjà.

Pour terminer, le mot de la fin te revient Julie. Tu as célébré tous mes *eurêkas*, mais avant ça, tu étais surtout présente dans les moments de doute pour me soutenir. Ta patience, ton amour et tes encouragements m'ont permis de traverser les moments les plus difficiles de cette thèse.

Résumé

Contexte. La transformation numérique de l'industrie, grâce à des technologies telles que les systèmes cyber-physiques, oblige les entreprises à améliorer la continuité numérique au sein de leur organisation en renforçant la capacité de leurs systèmes distribués à communiquer et à coordonner leurs activités, afin d'assurer des tâches complexes, telles que la planification intelligente, la détection d'anomalies, ou encore, la réduction de l'empreinte écologique des processus industriels.

Problème. Dans un environnement numérique instable et fortement hétérogène, en constante évolution tant sur le plan technique avec l'émergence de nouvelles technologies que sur le plan organisationnel avec les exigences accrues en matière d'agilité et d'adaptation des chaînes de production, l'établissement et le maintien de la communication, c'est-à-dire de l'interopérabilité entre les systèmes, est une tâche cruciale.

Proposition. Les transformations de modèles, pierre angulaire de l'Architecture Dirigée par les Modèles, pourraient apporter une solution concrète aux exigences d'une communication dynamique et durable entre les systèmes, où la transformation *modèle vers modèle* agit comme une fonction d'interopérabilité entre les modèles, et donc, entre les systèmes. La création et la maintenance des modèles de transformation constituent un véritable défi. Un nouveau paradigme utilisant des techniques d'apprentissage automatique pourrait simplifier le processus de création et de maintenance des modèles de transformation en apprenant, automatiquement, les règles de transformation entre les modèles.

Contributions. (1) Une application des principes de *l'Apprentissage par Renforcement*, plus particulièrement de *l'apprentissage Q*, est utilisée pour dériver automatiquement des règles de transformation et apprendre une transformation de modèles réutilisable en tant que fonction d'interopérabilité entre les métamodèles ; (2) Étant donné que les approches d'interopérabilité basées sur les modèles de transformation ne disposent pas d'un véritable cadre expérimental pour évaluer et comparer leur efficacité par rapport aux exigences de l'Industrie 4.0, un protocole expérimental est proposé. Le protocole expérimental développé permet de comparer les approches existantes et futures. L'objet d'étude de ce protocole porte sur les méthodes de construction des modèles de transformation, avec un focus particulier sur les approches d'apprentissage. Le protocole contient une grille de caractérisation des modèles de transformation, des mesures de performance et des jeux de données caractérisés.

Résultats. Un *benchmark* des approches existantes conclut ce travail, montrant l'efficacité des techniques d'apprentissage par renforcement dans l'apprentissage des règles de transformation qui relient deux langages de modélisation différents. Les résultats ont été validés à l'aide du protocole expérimental précédent. L'étude a prouvé que l'approche proposée est capable de dériver les règles

de transformation des jeux de données *Families2Persons* et *Class2Relation* sans perte sémantique, avec un temps d'apprentissage de quelques secondes, tout en minimisant l'effort humain requis pour construire le modèle de transformation.

Mots-clés : Industrie intelligente, Continuité numérique, Interopérabilité des systèmes, Interopérabilité sémantique, Transformation des modèles, Architecture Dirigée par les Modèles, Interopérabilité Dirigée par les Modèles, Apprentissage par renforcement.

Abstract

Context. The digital transformation of industry through technologies such as Cyber-Physical Systems is forcing companies to improve digital continuity within their organization by enhancing the ability of their distributed systems to communicate and coordinate their activities to ensure complex tasks, such as smart planning, anomaly detection, and ecological footprint reduction.

Problem. In an unstable and highly heterogeneous digital environment that is constantly evolving both technically with the emergence of new technologies and organizationally with the increased demands for agility and adaptation of production chains, establishing and maintaining communication, i.e. interoperability between systems, is a crucial task.

Proposal. Model transformations (MT), the cornerstone of Model-Driven Architecture (MDA), could provide a concrete solution to the requirements of dynamic and sustainable communication between systems, where *model-to-model* transformation acts as an interoperability function between models, and thus, between sytems. Creating and maintaining MT is a real challenge. A new paradigm using machine learning techniques could simplify the creation and maintenance of MT by automatically learning transformation rules between models.

Contribution. (1) An application of the *Reinforcement Learning* (RL) principles, in particular *Q-learning* , is used to automatically derive transformation rules and learn reusable model transformation as interoperability functions between metamodels ; (2) Since interoperability approaches based on MT do not have a true experimental framework to evaluate and compare their effectiveness with respect to Industrie 4.0 requirements, an experimental protocol is proposed. The developed experimental protocol allows comparison of existing and future approaches. The object of study of this protocol is the methods for building transformation models, with a particular focus on learning approaches. The protocol contains a characterization grid for transformation patterns, performance measures, and characterized datasets.

Results. A benchmark of existing approaches concludes this work, showing the effectiveness of reinforcement learning techniques in learning the transformation rules that connect two domain modelling languages. The results were validated using the following experimental protocol. This showed that the approach is able to derive the transformation rules of the Families2Persons and Class2Relation datasets without semantic loss, with a training time of 2 seconds, while minimising the human effort required to build the transformation model.

Keywords : Smart manufacturing, Digital continuity, System interoperability, Semantic interopera-

bility, Model transformation, Model-driven architecture, Model-driven interoperability, Reinforcement learning.

Table des matières

Liste des tableaux

Table des figures

Introduction générale

Contexte. Le monde est constamment en mouvement, les pratiques changent, les mentalités évoluent et de nouvelles technologies ne cessent d'émerger. Des mutations d'ordre technologique, organisationnelle et économique imposent aux entreprises des changements radicaux dans leur façon de concevoir et de produire afin de répondre aux grands enjeux environnementaux et sociaux de notre ère. Cette pression permanente s'articule par la nécessité de proposer des services et des produits à la demande, variés et personnalisables, selon les besoins, tout en respectant des exigences de qualités, de coût et des délais de développement toujours plus courts.

Ces exigences impliquent inévitablement d'accroître l'efficacité, la réactivité, la flexibilité (à savoir l'adaptabilité et la reconfigurabilité), ainsi que l'évolutivité des systèmes de production. Du point de vue de l'industrie, les critères d'efficacité doivent répondre à la fois à une dimension économique par une diminution des activités à non-valeur ajoutée, par l'optimisation des processus de production, et par la détection puis la réduction des non-conformités ; mais aussi par une dimension environnementale grâce à la diminution de l'empreinte écologique de ses activités. L'implication du consommateur dans les chaînes de décision lors du développement d'un produit nécessite d'accroître la réactivité par laquelle ses expériences, ses exigences et ses besoins seront pris en considération. La personnalisation des produits ou encore la mouvance des marchés requièrent de renforcer la flexibilité des systèmes de production, de sorte à s'adapter à de futurs tendances et besoins. Quant aux critères d'évolutivité, les systèmes de production doivent être en mesure d'intégrer de nouvelles technologies ou encore d'agrandir leur organisation. L'environnement industriel devient alors de plus en plus complexe et bouleverse radicalement la capacité de l'humain à superviser et contrôler l'intégralité de ses activités.

En réponse aux profonds changements de l'écosystème industriel, l'Industrie 4.0 s'impose comme la $4^{\text{ème}}$ révolution industrielle, en introduisant des concepts et des technologies clés, sous couvert d'une digitalisation poussée des processus et des systèmes physiques. L'Industrie 4.0 propage la vision d'une industrie intelligente et connectée à travers l'utilisation de technologies innovantes dans les domaines de l'information, de la communication et de l'intelligence artificielle. Les fondations de l'industrie intelligente reposent essentiellement sur la capture puis l'exploitation intensive des données produites tout au long du cycle de vie d'un produit. Une fois acquises, les données sont transformées en informations et en connaissances exploitables et réutilisables, pour assister la prise de décision par l'humain et la machine, afin d'optimiser le flux de production, d'assurer une planification intelligente ou encore de simuler et de superviser en temps réel les processus industriels.

La clé de voûte vers une industrie compétitive en phase avec les exigences de l'Industrie 4.0 s'ap-

puie sur des mécanismes de prise de décisions éclairées qui reposent eux-même sur des mécanismes d'agrégation, de corrélation ou encore de combinaison de données provenant de différentes sources d'informations. La continuité numérique et la convergence des données n'ont jamais été autant plébiscitées par les entreprises. La continuité numérique garantit la disponibilité d'une information fiable distribuée à la bonne personne et au bon moment, tandis que la convergence des données s'appuie sur l'idée que le traitement de données combinées est bien plus précieux que le traitement de chacune des données séparément.

Derrière ces deux dernières notions réside la nécessité de développer des mécanismes de connexion entre les systèmes, pour soutenir la mise en relation et la propagation des données, afin de promouvoir l'émergence d'une industrie intelligente, connectée et dirigée par les données. Le mécanisme de connexion mis en évidence pour assurer la communication entre des systèmes fait écho à la notion d'interopérabilité.

Problèmes. L'interopérabilité des systèmes est une condition *sine qua non* dans l'élaboration de la continuité numérique et de la convergence des données. La transformation digitale impose aux entreprises d'accroître l'interopérabilité des systèmes au sein de leurs organisations, en améliorant la capacité des systèmes distribués à communiquer pour coopérer puis coordonner leurs activités, afin de réaliser des tâches complexes qu'un système seul ne pourrait résoudre. L'ensemble des systèmes intervenant tout au long de la chaîne de valeur devrait être en mesure de communiquer de sorte à assurer l'échange et le partage de données entre les différents métiers d'une entreprise. Par ailleurs, la mondialisation et la sous-traitance d'activités accentuent le besoin de communication entre des organisations dispersées à travers le monde.

Malgré le besoin accru de connecter les systèmes et les organisations, l'environnement numérique actuel des entreprises semble moins propice à accueillir un climat d'hyperconnectivité. L'hétérogénéité des systèmes est un véritable frein à la construction d'un mécanisme de connexion pour l'interopérabilité. La grande diversité et disparité des supports de l'information entrave la capacité des systèmes à communiquer et nécessite systématiquement l'édification d'adaptateurs spécifiques à chaque communication pour garantir l'échange de données. Cependant, ces adaptateurs statiques, difficilement maintenables et peu évolutifs, sont inadaptés dans un environnement dynamique et évolutif tel que celui proposé par l'Industrie 4.0.

Objectifs. L'étude proposée dans ce manuscrit vise à développer, puis valider dans un cadre expérimental, un mécanisme de connexion qui assure la fonction d'interopérabilité des systèmes, tout en répondant aux exigences spécifiées par l'Industrie 4.0, à savoir, la création d'un mécanisme générique, automatique et dynamique.

Méthodologie de recherche. Les travaux d'Hevner [80] soulignent l'importance d'ancrer la recherche académique dans un environnement pratique pour apporter des solutions conformes aux attentes formulées par des utilisateurs. La recherche s'appuie alors sur des méthodologies rigoureuses pour développer, puis évaluer, les solutions proposées, de sorte à rester pertinente vis à vis des besoins initiaux, tout en évitant les dérives.

De ce fait, les Chapitres 1, 2, 3 et 4 seront centrées sur la formulation et la résolution d'un problème

de recherche en phase avec les exigences apportées par l'environnement étudié ; le Chapitre 5 portera, quant à lui, sur des exemples d'implémentation, ou passage à l'échelle des travaux réalisés en laboratoire dans l'environnement initial.

La méthodologie utilisée pour développer, puis valider la propostion présentée dans ce manuscrit, suivra les spécifications définies par la communauté GIS S.mart, concernant l'édification et la structuration d'une étude comparative de type *benchmark*. Dans leur étude, Pinquié *et al.* [135] proposent un métamodèle pour formaliser l'agencement des différents paramètres qui constituent un *benchmark* (voir Figure 1). Une instanciation de ce métamodèle sera construite tout au long du manuscrit et aura pour objectif d'apporter un cadre propice à la validation et à la comparaison entre les travaux présentés et l'état de l'art à l'égard des exigences formulées par l'environnement.

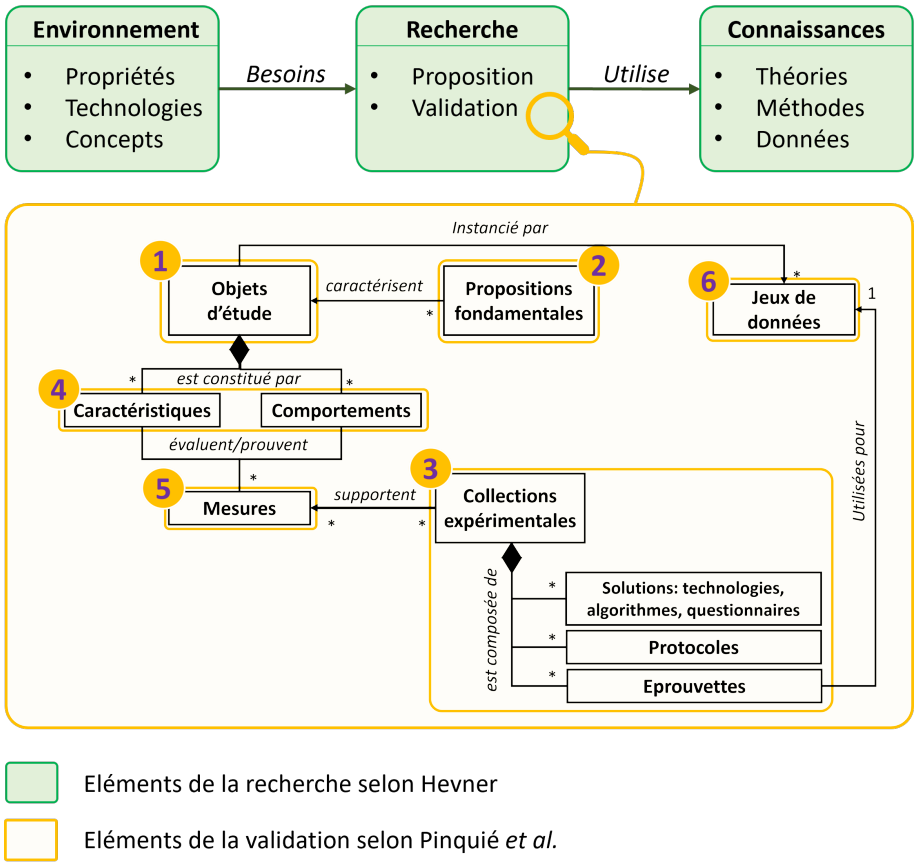


FIGURE 1 – Métamodèle d'une étude comparative de type *benchmark*

L'environnement, tel que définit par Hevner, correspondra dans cette étude à l'environnement numérique et physique d'une industrie à l'ère de la 4ème révolution industrielle. Au regard de la Figure 1, le manuscrit sera structuré de la façon suivante :

- *Chapitre 1.* L'environnement numérique et physique de l'Industrie 4.0, ainsi que les besoins qui en découlent, telle que l'importance d'assurer l'interopérabilité des systèmes, seront abordés. Les barrières et les freins technologiques qui restreignent et complexifient la construction d'un mécanisme

de connexion pour l'interopérabilité des systèmes seront ensuite exposés. L'analyse des propriétés, des technologies et des concepts propres à l'environnement de l'Industrie 4.0 permettra d'identifier **l'objet d'étude (1)** sur lequel porteront nos travaux et d'émettre une **proposition fondamentale (2)** en lien avec les besoins énoncés par l'Industrie 4.0.

- *Chapitre 2.* Une **collection expérimentale (3)** dédiée à l'objet d'étude défini précédemment sera construite à partir de l'état de l'art de la littérature et permettra une compréhension approfondie de l'état actuel des connaissances. Une discussion sera ensuite menée pour identifier les limites des approches existantes par rapport aux besoins qui émanent de l'Industrie 4.0.

- *Chapitre 3.* Au regard de l'analyse de l'état de l'art réalisée au chapitre précédent, les limites identifiées permettront de mettre en évidence des opportunités et des perspectives d'amélioration. Le mécanisme de connexion sur lequel porte ce manuscrit sera alors intégralement présenté. Afin d'évaluer et de comparer les performances de l'approche proposée vis-à-vis des performances des approches identifiées dans la collection expérimentale, un protocole de tests a été élaboré. Le protocole proposé contient une grille permettant d'exprimer les **caractéristiques et les comportements (4)** de l'objet d'étude, et des **indicateurs de performances (5)** permettant de mesurer à la fois les caractéristiques de l'objet d'étude et la pertinence des approches à répondre aux besoins de l'environnement. Enfin, des **jeux de données (6)** suffisamment représentatifs de l'environnement industriel seront exploités pour valider les approches de la collection expérimentale.

- *Chapitre 4.* Suivant le protocole de tests défini précédemment, une étude comparative de type *benchmark* sera menée pour déterminer les limites et les avancées atteintes par le mécanisme de connexion proposé, vis-à-vis des approches présentes dans la collection expérimentale. Les résultats et les preuves de validation obtenus seront présentés en détail dans ce chapitre.

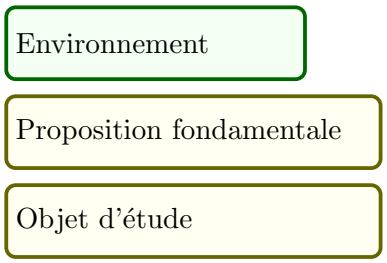
- *Chapitre 5.* Le passage à l'échelle est une étape non négligeable pour valider la cohérence de l'approche proposée au regard des besoins énoncés par le contexte de l'étude. Pour cela, des cas d'études concrets, soumis aux contraintes et aux aléas de l'Industrie 4.0 seront présentés.

Chapitre 1

Environnement d'étude, problématique, proposition

Contenu

Suivant le cadre de recherche instauré par Hevner [80], l'analyse approfondie de l'environnement d'étude est primordiale pour en déduire les besoins qui en découlent. L'objectif de ce premier chapitre est d'introduire les propriétés, les technologies ainsi que les concepts sur lesquels repose l'environnement numérique et physique de l'Industrie 4.0. Des verrous technologiques seront alors mis en évidence et permettront d'identifier l'objet d'étude sur lequel portera ce manuscrit. Une proposition fondamentale sera ensuite émise afin d'apporter une solution concrète pour résoudre les limites de l'environnement étudié.



1.1 Environnement d'étude : l'Industrie 4.0

L'intégration des outils du numérique dans le quotidien des acteurs de l'industrie est devenue un enjeu incontournable pour la productivité et la compétitivité des entreprises du secteur manufacturier. Des changements radicaux découlent de la transformation numérique de l'industrie. Ce nouveau paradigme se nomme Industrie 4.0 et s'impose comme la $4^{\text{ème}}$ révolution industrielle[104].

1.1.1 Ses propriétés

L'Industrie 4.0 propage la vision d'une industrie *connectée*, où la communication et la collaboration entre les machines, les systèmes numériques et l'humain est rendu possible sans aucune restriction [174] ; *Intelligente* où l'optimisation des chaînes de production se traduit par la capacité des systèmes à s'adapter et se reconfigurer pour faire face à de nouveaux besoins et changements [198] ; *Dirigée par les données* grâce à des mécanismes de prise de décisions rationnels qui s'appuient sur l'analyse en temps réel des données [161]. La virtualisation de l'industrie promet des opportunités tant sur le plan organisationnel et économique en améliorant la modularité des systèmes, la flexibilité des chaînes de production, ou encore la supervision des processus [41], que sur le plan écologique en diminuant l'empreinte environnementale des processus industriels du secteur manufacturier [65, 12].

Dans leur étude Hermann *et al.* [79] décrivent les principales propriétés qui spécifient l'architecture générale de l'Industrie 4.0. Le Tableau 1.1 résument ces propriétés.

L'industrie 4.0 repose essentiellement sur la collaboration intensive de ses systèmes physiques et numériques, qui, grâce à la décentralisation de la prise de décision, sont capables de s'auto-ajuster pour accomplir des activités conjointes.

Propriétés	***Description***
L'interopérabilité	Étroitement liée à la notion d'industrie connectée, l'interopérabilité permet d'établir un climat propice à l'échange et au partage des données entre les systèmes, grâce à des mécanismes de connexion. L'interopérabilité est une propriété précieuse de l'Industrie 4.0. Elle est à l'origine de la continuité numérique au sein et entre les organisations, ainsi que de la convergence des données entre les systèmes. L'Industrie intelligente émane de la capacité des systèmes et des organisations à communiquer et à coopérer.
La virtualisation	La virtualisation des processus industriels, des chaînes d'approvisionnement (*supply-chain*), ou encore des produits eux-même, permet une supervision et des analyses en temps réel de l'encours des activités pour l'ensemble des participants de la chaîne de valeur. La virtualisation simplifie la communication entre les différents parternaires industriels, et améliore la productivité en procurant des analyses en temps réel basées sur les données [28].
La décentralisation	La propriété de décentralisation profère une autonomie de prise de décision aux systèmes de l'entreprise. Les décisions sont ainsi prises en fonction du contexte et de l'état dans lequel se trouve le système grâce à l'acquisition en temps réel des données. La décentralisation des prises de décisions permet une meilleure collaboration entre les systèmes (physiques ou numériques) et améliore efficacement la flexibilité des chaînes de production [123].
Temps réel	La propriété de temps réel s'articule autour de la nécessité de collecter et d'analyser les données, pour produire un suivi des processus ou un support à la décision en temps réel. L'accès à des données constamment actualisées permet d'améliorer drastiquement les capacités de réactivité et d'anticipation des entreprises face à des évènements en-cours ou qui vont avoir lieu.
Architecture Orientée Service	Étroitement liée à la virtualisation des entreprises, la servitisation de l'Industrie 4.0 répond à une forte demande de personnalisation des produits [63]. Par ailleurs du point de vue de la performance industrielle, une Architecture Orientée Service (SOA) a la particularité de disposer d'une grande flexibilité dans sa capacité à coupler, à travers des interfaces, des services autonomes et réutilisables. De cette façon, les fonctionnalités des systèmes sont encapsulées dans des services qui peuvent être utilisés à tout moment, en fonction du besoin et de la demande, de sorte à répondre à de nouvelles spécifications.
La modularité	La modularité d'une architecture permet d'absorber les changements (ajout, remplacement, suppression ou encore évolution des systèmes) et de coupler aisément des systèmes entre eux. Une architecture modulaire peut être facilement ajustée pour ingérer de nouvelles fonctionnalités.

TABLE 1.1 – Propriétés de l'Industrie 4.0

1.1.2 Ses technologies

Des technologies clés aux interstices des domaines de l'information, de la communication et de l'intelligence artificielle constituent la base des fondations de l'Industrie 4.0. Ces technologies ont pour

objectif d'apporter des solutions concrètes pour répondre aux propriétés identifiées par Hermann *et al.* [79] et favoriser l'émergence de l'industrie intelligente.

1.1.2.1 Les Systèmes Cyber-Physiques

L'industrie intelligente repose sur la capacité des systèmes, des processus industriels, et des entreprises (fournisseurs et partenaires commerciaux) à s'interconnecter de manière à coordonner, à l'unisson, leurs activités, de sorte à accomplir un objectif commun. Lorsque plusieurs systèmes travaillent en collaboration pour réaliser des activités conjointes, cette composition de systèmes peut être considérée comme un seul et unique système dit "*complexe*" [131]. Les *Systèmes Cyber-Physiques (CPS)* sont considérés comme des systèmes complexes qui intègrent à la fois des systèmes physiques (mécaniques, électroniques...) et numériques (algorithmes, logiciels...). Par définition, un CPS est considéré comme un système de systèmes (*System of Systems* (Sos)) [115]. En d'autres termes, il s'agit d'un système qui intègre différents sous-systèmes autonomes, coopérant pour réaliser des activités communes. Leur synergie permet d'accomplir des objectifs de plus haut niveau (*e.g.* optimisation de la production, planification intelligente), dont le résultat ne se traduit pas forcément par la somme d'activités réalisées individuellement [62].

Plus particulièrement, les CPS sont composés de trois éléments principaux : (1) l'acquisition de données en temps réel depuis l'environnement physique, puis la transmission de ces données vers l'environnement numérique ; (2) le traitement, ou conversion des données transmises en informations utiles ; (3) l'auto-reconfiguration ou l'adaptation des systèmes en fonction de l'analyse cognitive des informations [105]. Ces trois éléments résument l'architecture 5C proposée par Kao *et al.* [90] : (1C) la *connexion* entre les systèmes physiques et numériques pour promouvoir l'échange de données ; (2C) la *conversion* des données en informations utiles et exploitables ; (3C) le niveau *cybernétique* permet la construction d'une représentation numérique, ou avatar des systèmes physiques ; (4C) la *cognition*, où des mécanismes d'analyse et de raisonnement appliqués aux données sont exploités pour assister la prise de décision ; (5C) la *configuration* qui donne la capacité aux systèmes de s'auto-reconfigurer pour répondre à de nouvelles fonctionnalités, s'auto-ajuster pour réguler leur charge de travail et planifier de nouvelles activités, et s'auto-optimiser pour réduire les non-conformités et les délais du cycle de production. Par l'intermédiaire des CPS, l'industrie intelligente transite d'une architecture centralisée vers une industrie décentralisée où la décision revient aux systèmes.

Les interactions entre les environnements physique et numérique sont rendues possible grâce à des mécanismes de connexion, qui garantissent la convergence des données issues des systèmes physiques et des systèmes numériques [124]. Les activités de l'environnement physique sont alors supervisées, contrôlées ou encore coordonnées depuis l'environnement numérique [140]. Le concept de CPS pousse l'automatisation des processus dans ses derniers retranchements.

Pour pleinement mesurer la puissance des CPS, il est nécessaire de mettre en lumière les propriétés qui les définissent. Etant donné que les CPS sont des cas particuliers de SoS, ils héritent donc de leurs propriétés. [62] ont menés une investigation complète pour déterminer les propriétés qui caractérisent un SoS. Leur étude s'appuie sur les précédent travaux réalisés par [21, 125]. Le Tableau 1.2 présente ces propriétés :

Propriétés	***Description***
L'autonomie	Chaque sous-sytème est initialement conçu pour répondre à des fonctionnalités précises. Ils répondent à leurs propres règles et restent libres et indépendants. Leurs règles ne peuvent être transgressées par un autre sous-système.
La diversité	La nature hétérogène du systèmes dans son ensemble, permet d'intégrer des sous-systèmes qui n'emploient ni les mêmes technologies, ni les mêmes standards, ni les mêmes représentations conceptuelles des données.
L'évolutivité	Les sous-systèmes sont sujet à des modifications courantes liées à l'évolution de leur environnement. Ces changements prennent la forme de nouvelles fonctionnalités à intégrer ou à retirer du sous-système.
La dynamicité	La propriété de dynamicité fait référence à la capacité du système à absorber les changements. La composition d'un SoS peut être impactée par des opérations d'ajout et de supression de sous-systèmes qui ne doivent pas pour autant compromettre le fonctionnement global du système.
La connexion	Des mécanismes de connexion permettent la communication entre les sous-systèmes.
L'interdépendance	Par l'intermédiaire des mécanismes de connexion qui garantissent l'échange de données entre les sous-systèmes, l'interdépendance est la capacité des sous-systèmes à combiner leurs efforts pour répondre à un besoin de manière collective .
L'émergence	La naissance d'un comportement émergeant se traduit par l'ajustement et la combinaison des activités des sous-systèmes de sorte à résoudre des problèmes complexes auxquels ils ne peuvent apporter de réponse individuellement.

TABLE 1.2 – Propriétés des Systèmes de Systèmes

Ces propriétés définissent un CPS comme un système en perpétuel évolution, qui peut englober de nouveaux systèmes inconnus jusqu'à présent, s'auto-reconfigurer pour améliorer ses performances et résoudre collectivement de nouveaux problèmes, sans jamais compromettre l'individualité de ses sous-systèmes. Selon Abbott [1], une architecture de type SoS est (1) "*ouverte par le haut*", la structure d'un SoS n'est jamais fixe et est toujours propice à l'addition d'un nouveau système ; (2) "*ouverte par le bas*", des communications spécifiques entre les sous-systèmes peuvent évoluer ou être ajouter ; (3) "*en évolution constante*", un SoS n'est jamais complet et évoluera toujours en réponse aux modifications qui ont lieu dans son environnement.

1.1.2.2 Les objets connectés

Les objets connectés de l'*Internet of Things (IoT)* sont une composante indissociable des CPS. Ils permettent l'acquisition de données, depuis l'environnement physique à travers l'utilisation de capteurs. Inversement, les IoT tels que les actionneurs et les machines, agissent dans l'environnement physique, tout en étant contrôlés depuis l'environnement numérique. L'*Internet of Everything (IoE)* est un concept qui étend l'idée initiale de l'IoT, en intégrant non seulement les appareils de type capteurs et actionneurs, mais aussi les personnes, les processus, les services et les données. L'IoE est un réseau

d'objets, d'individus et de systèmes interconnectés, qui communiquent et partagent des données pour collaborer les uns avec les autres. Dans le contexte de l'Industrie 4.0, les objets connectés, les machines, les processus et les lignes de production, ainsi que les humains sont interconnectés tout au long de la chaîne de valeur d'une entreprise [167]. Dans un contexte d'industrie intelligente et connectée, trois propriétés caractérisent les IoT : (1) la propriété de contexte permet aux objets de capturer l'état dans lequel ils se trouvent (localisation, conditions physiques...) ; (2) la propriété d'omniprésence met en évidence la notion de communication à grande échelle entre les objets qui constituent le réseau ; (3) la propriété d'optimisation exprime la capacité des objets à s'adapter selon un état donné [184].

> *Cisco*[a] *defines the Internet of Everything (IoE) as the networked connection of people, process, data, and things. The benefit of IoE is derived from the compound impact of connecting people, process, data, and things, and the value this increased connectedness creates as "everything" comes online.*
>
> a. `https://www.cisco.com/`

1.1.2.3 Le *Big Data*

Les données d'une entreprise ont une valeur inestimable [38]. L'analyse des données par les technologies du *Big Data* permet d'optimiser les performances d'un système individuellement et collectivement, et à plus grande échelle, les performances de toute une ligne de production. Des informations précieuses sont extraites de cette masse volumineuse de données. L'industrie intelligente repose sur des mécanismes d'analyse, en temps réel, des données provenant de diverses sources d'information, pour assister la prise de décision. Ces mécanismes d'analyse reposent essentiellement sur les dernières avancées des domaines de *l'apprentissage machine* et de *l'apprentissage profond* [175].

Le *Big Data* est une source d'opportunités considérables pour les entreprises. L'intégration de diverses sources d'informations, grâce aux mécanismes de connexion qui relient les systèmes entre eux, permet d'accéder à un réel contexte de *Big Data*, où les données sont corrélées et utilisées pour prendre des actions concrètes en temps réel [94]. La collecte, puis l'analyse de cette masse de données, permet d'améliorer la productivité et l'efficacité des entreprises. Avoir accès aux données archivées permet par exemple de prédire la défaillances des machines [94], ou encore, d'optimiser les plans de production en fonction des ressources disponibles [159].

1.1.3 Ses concepts

Les données jouent un rôle *sine qua none* dans l'émergence de l'industrie intelligente. Elles sont au cœur des grands changements attendus par l'industrie. L'industrie dirigée par la connaissance et le savoir humain migre peu à peu vers une industrie dirigée par les données, paradigme incontournable dans l'élaboration d'une industrie dite intelligente. La notion "*d'intelligence*" réfère à la capacité de l'industrie à capter de la donnée tout au long du cycle de vie d'un produit, pour ensuite la transformer en informations et en connaissances exploitables et réutilisables grâce aux technologies du *Big Data*,

dans le but d'adapter et d'optimiser le flux de production, de simuler et de superviser en temps réel les processus industriels et d'assister la prise de décision par l'humain et la machine dans leurs activités [109]. La prise de décision, dans un environnement complexe qui fait intervenir une multitude de facteurs (humains, métiers, processus, produits et organisation...), s'appuie alors sur l'analyse de données tangibles, qui émanent directement de différentes sources d'information [138].

Deux concepts importants émanent de la notion d'industrie intelligente : la convergence des données et la continuité numérique.

1.1.3.1 La convergence des données

Le principe de convergence des données transcende les silos traditionnels de la connaissance. Par essence, la notion de convergence nécessite la mise en relation des données provenant de domaines d'activités différents et de sources d'information très diverses, pour en tirer des informations et des connaissances plus approfondies. Relier des sources d'information initialement isolées permet d'aboutir à un réel contexte de *Big Data*, où la corrélation et la combinaison des données garantissent une vue complète et cohérente de l'information. De ce fait, le concept de convergence des données repose sur l'idée que des données isolées et analysées séparément sont nettement moins précieuses que des données combinées et analysées ensemble. *In fine*, l'agrégation des données permet d'identifier des relations, des tendances et des modèles qui seraient indétectables si les silos de données étaient analysés séparément.

Parmi les technologies imaginées par l'Industrie 4.0, les CPS sont à l'origine de la convergence des données provenant des environnements physique et numérique de l'industrie, grâce à l'exploitation intensive des objets connectés de l'IoT. Les technologies du *Big Data* et de l'intelligence artificielle appliquées aux données permettent d'assister la prise de décision dans le cadre des maintenances prédictives [25] ou encore d'optimiser la planification et le contrôle de la production [31].

1.1.3.2 La continuité numérique

Par définition, la continuité numérique fait référence à un flux d'informations ininterrompu entre toutes les parties prenantes d'une chaîne de valeur. Les données proviennent des différentes phases du cycle de vie d'un produit, de sa conception à sa fabrication, en passant par son utilisation et sa maintenance. Disposer d'un accès sans restriction à une information fiable, constamment réactualisée, adressée aux bonnes personnes et aux bons moments est l'essence même de la continuité numérique.

La continuité numérique et la convergence des données sont deux concepts indissociables. La continuité numérique facilite la transmission d'informations au sein d'une organisation, tandis que la convergence des données permet d'exploiter pleinement le patrimoine informationnel des entreprises. Pour assurer ces deux phénomènes, l'ensemble des systèmes physiques et numériques doit être en mesure de communiquer. Le mécanisme de connexion, qui permet la communication entre les différents systèmes, fait partie des concepts clés de l'Industrie 4.0.

1.1.4 Ses besoins

La communication entre les objets connectés, les machines et les systèmes d'information est un prérequis indispensable pour assurer l'émergence des technologies telles que les CPS, pour assurer la vision d'une industrie où le flux d'informations est continu et la convergence des données possible. Il est essentiel d'interconnecter les différents systèmes de la production, de sorte à promouvoir leur collaboration. Selon Tao *et* Qi [160] le mécanisme de connexion mis en évidence peut être analysé sous trois perspectives :

- *physique-vers-physique (P2P)*, à savoir une connexion entre les équipements de la production, tels que les objets connectés et les machines ;
- *numérique-vers-numérique (N2N)*, lorsqu'il s'agit d'interconnecter les systèmes d'information de l'entreprise tels que les progiciels de gestion intégré (ERP), les systèmes d'exécution de la fabrication (MES) ou encore les gestionnaires de données relatives aux produits (PDM) ;
- *physique-vers-numérique (P2N)*, comme c'est le cas lors de la mise en oeuvre d'un CPS, où les équipements sont connectés à un système d'information.

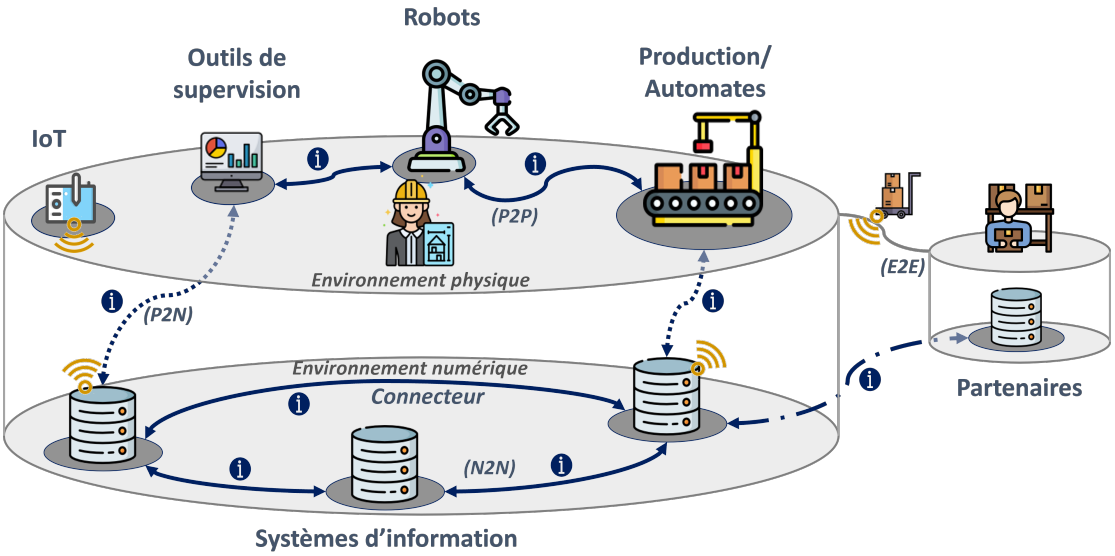


FIGURE 1.1 – Système cyber-physique

Par ailleurs, pour assouvir le besoin de réactivité et de flexibilité de l'industrie, ainsi que la continuité numérique au sein des organisations, l'intégralité des sources d'informations, qu'elles soient physiques ou numériques, doit être en mesure de communiquer et d'échanger des données tout au long de la chaîne de valeur. Avec la mondialisation et la décentralisation des activités, cela nécessite inévitablement d'inclure les systèmes des fournisseurs, des partenaires commerciaux, des fabricants d'équipement et des entreprises de logistique, pour garantir la communication et la collaboration entre plusieurs organisations distinctes. L'interconnexion de l'ensemble des intervenants, tout au long de la chaîne de valeur, est un enjeu primordial pour assurer la réactivité et la flexibilité de toute une chaîne de production. La Figure 1.1 illustre les différents systèmes ainsi que les différentes connexions qui

peuvent intervenir dans le contexte d'un CPS.

La prédisposition des systèmes à communiquer, échanger de l'information puis utiliser les informations échangées, fait appel à la notion *d'interopérabilité des systèmes* [35].

> *Industry 4.0 can be defined as the industrial vision to enable people and things to be connected* ***Anytime****,* ***Anyplace****, with* ***Anything*** *and* ***Anyone*** *ideally using* ***Any*** *network and* ***Any*** *service [174].*

1.1.4.1 L'interopérabilité des systèmes

L'interopérabilité est l'élément clé qui permet l'automatisation de l'Industrie 4.0. Différentes définitions de l'interopérabilité ont été proposées ces dernières années. Parmi elles, nous retiendrons la définition de l'*IEEE Standard Computer Dictionary*, selon laquelle l'interopérabilité est la capacité de deux ou plusieurs systèmes ou composants à échanger des informations puis utiliser les informations échangées [83]. Wegner étendra la précédente définition en rajoutant que les systèmes doivent coopérer malgré leurs différences en terme de langage de développement, d'interfaces et de plateforme d'exécution [179]. Les précédentes définitions induisent plusieurs niveaux d'interopérabilité :

- *Interopérabilité des entreprises.* Dans un contexte de mondialisation, où la chaîne de valeur intègre à la fois des partenaires commerciaux et des fournisseurs dispersés à travers le monde, l'interopérabilité des entreprises révèle la capacité des organisations et de leurs processus à être connectés au sein d'un réseau d'entreprises, pour assurer l'échange de données [37, 56].
- *Interopérabilité des processus.* L'environnement industriel est composé de différents processus constamment en interaction (en série ou en parallèle), qui font intervenir de nombreuses ressources (humaines, financières et matérielles). L'interopérabilité des processus consiste à relier différents processus pour former un processus collaboratif commun [35].
- *Interopérabilité des systèmes.* L'interopérabilité des systèmes, des équipements, des produits ou de tout autres objets, dont les interfaces sont intégralement connues, désigne leur capacité à fonctionner avec d'autres objets et ce, sans restriction d'accès ou de mise en oeuvre. Lors de l'interconnexion de deux objets, il est nécessaire de considérer l'interopérabilité technique et syntaxique.
  - *Interopérabilité technique.* L'interopérabilité technique permet la collaboration de différentes plateformes, indépendamment des spécifications fournies par les vendeurs, des langages de développement utilisés ou encore des protocoles de communication exploités (MQTT, http...).
  - *Interopérabilité syntaxique.* L'interopérabilité syntaxique est adressée à travers l'utilisation de formats de fichier dans lesquels les données sont transportées d'un système vers un autre. Parmi les formats d'encodage des données couramment utilisés, les standards *Extensible Markup Language (XML)* et JSON sont les plus représentés.
- *Interoperabilité des données.* La mise en relation des données provenant de différentes sources

d'informations fait intervenir deux notions : l'interopérabilité sémantique et l'interopérabilité des modèles.

- *Interopérabilité sémantique.* Lorsque l'IEEE définit l'interopérabilité comme l'échange puis l'utilisation des données échangées, la notion de compréhension des données échangées est implicitement sous entendue. La compréhension du sens des données fait référence à la définition de l'interopérabilité sémantique, qui se traduit par l'interprétation automatique des données sans ambiguïté de sens [139]. De cette façon, le consommateur et le fournisseur de données ont une compréhension commune de la signification des données échangées [78].
- *Interopérabilité des modèles.* L'interopérabilité des modèles consiste à tisser des relations, ou mappe (*mapping* en anglais), entre deux représentations conceptuelles des données. Ces relations spécifient les équivalences sémantiques entre deux modèles. Une fois établi, un mécanisme de traduction est utilisé pour convertir les données provenant du système fournisseur dans la sémantique du système qui va consommer les données [126].

Cependant, il est légitime de se demander si l'environnement numérique actuel des entreprises est prédisposé à répondre aux exigences accrues d'interopérabilité entre les systèmes. L'environnement numérique actuel des entreprises est-il réellement propice pour garantir la vision d'une Industrie 4.0 hyperconnectée ?

1.1.4.2 Les grandes approches pour l'interopérabilité des systèmes

Le standard ISO 14258 [1] préconise trois grandes approches pour établir un environnement numérique coopératif et promouvoir l'interopérabilité des systèmes : les architectures intégrées, unifiées et fédérées (voir Figure 1.2).

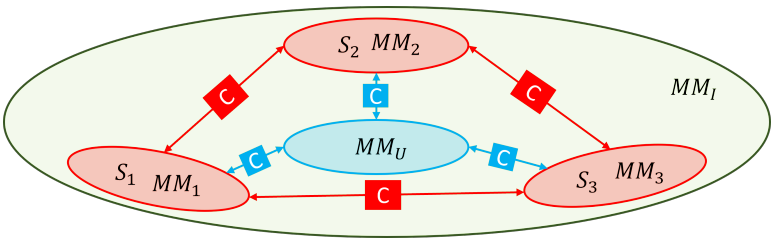


FIGURE 1.2 – Grandes approches pour l'interopérabilité des systèmes selon le standard ISO 14258

Selon la Figure 1.2 : En vert, l'approche intégrée expose un unique métamodèle intégré MM_I, qui exprime la sémantique et la structure des données de l'ensemble des systèmes S_1, S_2 et S_3. En bleu, l'approche unifiée emploie un ou plusieurs métamodèles unifiés MM_U, qui expriment des relations d'équivalence avec les concepts des métamodèles locaux MM_1, MM_2 et MM_3 des systèmes S_1, S_2 et S_3. En rouge, les systèmes communiquent 1-1 sans aucun intermédiaire, grâce à des relations sémantiques qui spécifient les concepts équivalents entre les différents métamodèles locaux.

Architecture intégrée. L'approche intégrée emploie une unique structure de données commune et partagée par l'ensemble des systèmes de l'entreprise. Les modèles de chacun des systèmes sont conçus

1. https://www.iso.org/obp/ui/fr/#iso:std:iso:14258:ed-1:v1:en

et interprétés en accord avec le modèle de référence. Cela signifie que les métamodèles MM_1, MM_2 et MM_3 sont cohérents vis-à-vis de la sémantique du métamodèle intégré MM_i.

Dans le cadre d'une architecture intégrée, les systèmes sont étroitement couplés (*tightly coupled* en anglais). Cela signifie que les relations entre les modèles sont sémantiquement explicites, puisqu'elles s'appuient sur une représentation des données partagée par tous les modèles. L'approche intégrée est d'avantage orientée sur l'intégration des systèmes, plutôt que sur leur capacité à interopérer. La notion d'intégration est caractérisée par l'incorporation, l'assimilation ou encore la combinaison d'un système au sein d'une architecture, qui impose le formalisme de ses données pour propager un cadre de compression commun [151]. Autrement dit, l'approche intégrée permet la migration d'un environnement hétérogène vers un environnement homogène.

Architecture unifiée. L'approche unifiée, quant à elle, adopte un métamodèle qui sert de référence, i.e. un métamodèle pivot, ou unifié MM_U, pour établir un mappage qui spécifie les équivalences sémantiques entre les concepts des métamodèles locaux. Ces relations sémantiques sont alors exploitées par un mécanisme de traduction qui permet de transformer, par l'intermédiaire du métamodèle unifié, les données d'un domaine spécifique vers un autre, en minimisant la perte d'information. Dans le cas d'une approche unifiée, le couplage entre les systèmes est dit "*lâche*" (*loosly coupled* en anglais). Cela signifie que les systèmes évoluent indépendamment et sont interconnectables via leurs interfaces.

Dans le cas d'une médiation sémantique, une ontologie de référence peut servir de modèle pivot pour garantir un cadre de compression commun aux systèmes, à travers la standardisation des concepts d'un domaine [122]. Ce référentiel décrit la signification des données dans un environnement précis (domaine spécifique) et procure les fondations propices au développement d'une interopérabilité sémantique entre les systèmes. La création d'une ontologie médiatrice assure alors un lexique commun compris et partagé par les systèmes, qui s'en servent pour tisser des relations sémantiques entre leurs modèles de données. La notion de partage indique qu'une ontologie capture des connaissances conceptuelles qui sont acceptées par un groupe de systèmes [43]. Les conflits sémantiques entre les données sont ainsi minimisés, puisque les systèmes partagent une représentation conceptuelle normalisée des données du domaine dans lequel ils opèrent.

Architecture fédérée. L'approche fédérée n'impose en revanche aucun médiateur et aucun système n'impose le sien. A la différence de l'approche unifiée, aucun métamodèle n'est prédéfini. Le couplage entre deux systèmes est fait à partir de leurs interfaces et l'échange de données est assuré par un mécanisme de traduction spécifique aux deux systèmes interconnectés. De ce fait, il existe $n * (n-1)$ traducteurs, où n correspond au nombre de systèmes à relier. L'un des défis avérés d'une architecture fédérée consiste à automatiser, à la demande, la création de mappages AAA (*Anybody-Anywhere-Anytime*) entre les systèmes directement à partir de leurs métamodèles.

1.1.4.3 Les grands défis de l'interopérabilité

La création de mécanismes de connexion pour promouvoir l'interopérabilité des systèmes soulève des défis considérables à résoudre. L'industrie 4.0, tout comme les CPS, présente l'interopérabilité des systèmes comme la brique de base à élaborer pour garantir l'émergence de l'industrie intelligente.

L'analyse de leurs propriétés a permis d'identifier les contraintes et les exigences que devront respecter les mécanismes de connexion pour pérenniser la vision d'une Industrie 4.0 connectée. La Figure 1.3 illustre la corrélation des propriétés de l'Industrie 4.0, définies par [79] avec les propriétés des CPS définies par [90]. Étant donné que les CPS sont des cas particuliers de SoS, les propriétés définies par [62] ont été rajoutés à l'analyse.

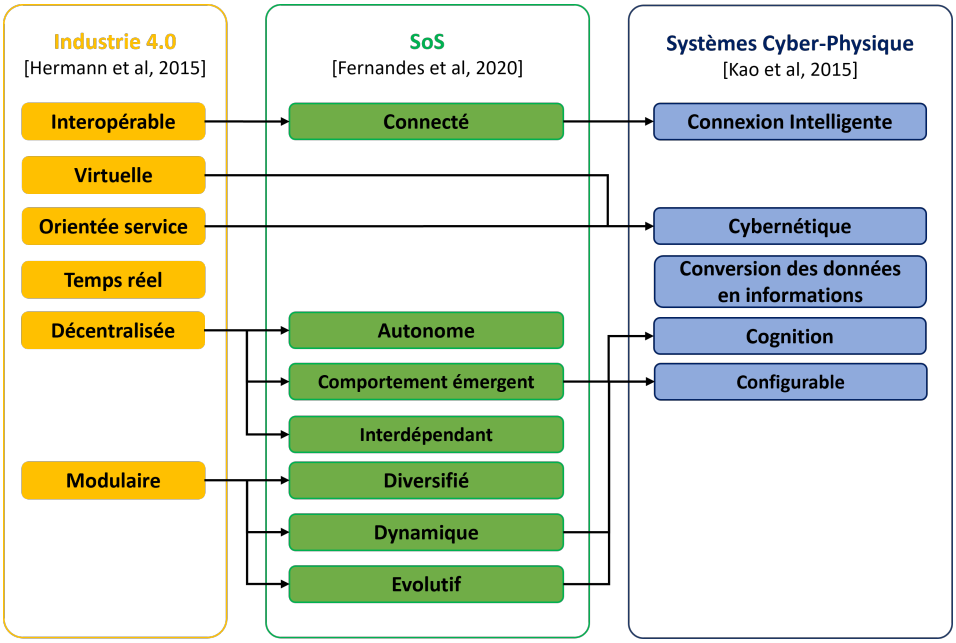


FIGURE 1.3 – Corrélation des propriétés de l'Industrie 4.0 avec les propriétés des SoS et des CPS

A partir de l'analyse précédente, des défis pour l'interopérabilité peuvent être formulés :

Un mécanisme générique. Les propriétés d'un SoS mettent en évidence la nécessité de prendre en considération la grande diversité des sous-systèmes à interconnecter. En effet, les CPS intègrent des systèmes très divers issus des environnements physique et numérique. De ce fait, le mécanisme de connexion doit être générique pour répondre à une large variété de types de communication.

Un mécanisme automatique. L'Industrie 4.0 est une industrie temps réel et modulaire. Cela signifie que les systèmes doivent être interconnectés à la volée. Le temps lié à l'interconnexion des systèmes doit être minimisé, de sorte à maximiser la réactivité lors de l'intégration d'un nouveau système dans la chaîne de production. Le mécanisme de connexion doit être automatique pour limiter les coûts et les délais liés à l'intégration des systèmes.

Un mécanisme dynamique. L'Industrie 4.0 propage la vision d'un environnement numérique flexible et évolutif. Les systèmes qui composent cet environnement sont totalement autonomes et libres d'être ajustés et reconfigurés. Le mécanisme de connexion doit être dynamique de façon à absorber les changements de l'environnement, et plus particulièrement les évolutions des systèmes.

1.2 Problématique

L'environnement numérique actuel des entreprises semble moins propice aux prérequis d'hyper-connectivité des technologies de l'Industrie 4.0, telles que les CPS [70, 132] et les objets connectés de l'IoT [128, 9]. Rendre interopérables des systèmes, qui initialement ne sont pas conçus pour travailler ensemble, est une tâche complexe et coûteuse. L'incompatibilité des systèmes et leur non-concordance à divers niveaux (technologique, sémantique...) inhibent la capacité des systèmes à communiquer et à échanger des données. Principalement, les conflits qui restreignent l'interopérabilité des systèmes proviennent de l'hétérogénéité des plateformes, des systèmes, des données et de leur représentation conceptuelle.

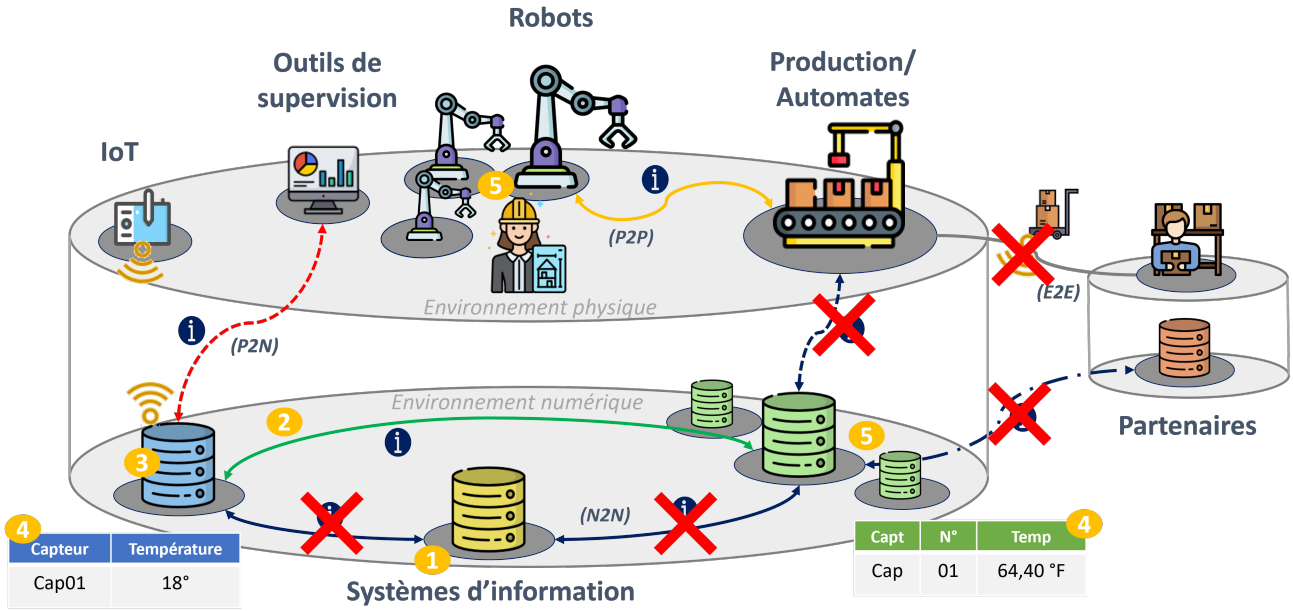


FIGURE 1.4 – La réalité numérique

Selon l'annotation de la Figure 1.4 : (1) les systèmes sont isolés et ne communiquent pas ; (2) les connecteurs, qui relient les systèmes entre eux, utilisent des protocoles et standards de communication différents, ainsi que des formats d'échanges spécifiques ; (3) les systèmes exploitent différentes plateformes, différents langages de programmation, ou encore différents systèmes d'exploitation ; (4) les représentations conceptuelles des données sont propres à chaque système ; (5) la démultiplication des sources d'information.

1.2.1 Un environnement hétérogène et instable

1.2.1.1 L'hétérogénéité des plateformes

Avec l'avènement des CPS et des IoT, de nombreuses plateformes proposées par Amazon (AWS IoT), Cisco (Jasper), IBM (Watson), Apple (HomeKit), Google (Brillo) et Microsoft (Azure IoT),

ou encore des plateformes *cloud* [141] ont rapidement proliféré. La démultiplication des plateformes encourage l'expansion d'un écosystème de platefromes incompatibles, dans l'incapacité de fonctionner ensemble. Chacune des plateformes met en évidence sa propre architecture logicielle employant des protocoles de communication, des formats d'échanges et des interfaces spécifiques ainsi qu'une sémantique des données particulière. Les plateformes promeuvent un environnement cloisonné, qui réduit à la fois leur capacité à s'interconnecter avec d'autres plateformes, mais également la capacité de certains IoT, incompatibles aux spécifications de la plateforme, à se connecter efficacement [128]. Des adaptateurs sont nécessaires pour permettre l'interopérabilité entre les plateformes à travers leurs interfaces.

Dans leur étude, Noura *et al.* [127] expriment la nécessité d'une interopérabilité entre les plateformes, qui permet la construction d'applications multiplateformes favorisant, l'accès aux données provenant de plusieurs plateformes spécifiques.

1.2.1.2 L'hétérogénéité des systèmes

De la même façon que les plateformes pour l'IoT, l'hétérogénéité des systèmes (systèmes d'information, équipements, objets connectés...), est en partie provoquée par la grande diversité des systèmes et des vendeurs qui les proposent. L'incompatibilité des systèmes favorise le cloisonnement de l'information et réduit drastiquement l'intégration de nouvelles sources de données. L'hétérogénéité des systèmes est causée, sur le plan technologique, par l'utilisation de protocoles de communication divers et de formats de données d'échange très variés :

- *Communication.* Différents standards et protocoles existent pour assurer la communication des systèmes. Parmi eux, le standard OPCUA [114], les protocoles de messagerie MQTT (*Message Queuing Telemetry Transport*) proposés par OASIS [2] et CoAP [23], ou encore le protocole de communication HTTP (*Hyper Text Transport*) pour le *World Wide Web* et les API *Restful* [143]. Plus particulièrement, concernant les IoT, les protocoles de communication 4G/5G, *bluetooth* et RFID sont communément employés.
- *Syntaxe (format) des données.* Une fois la communication établie, les messages échangés doivent être formulés dans un encodage, ou syntaxes particulières. La syntaxe peut être définie comme le format normalisé des données. En revanche, le format des données est totalement indépendant de la structure sémantique des données. Parmi les formats d'échanges de données, le langage XML (*Extensible Markup Language*) [27] est un langage basé sur l'utilisation de balises, le format JSON (*JavaScript Object Notation*) est une liste ordonnée des objets, le format CSV *Comma-Separated Values*) ou encore le langage STEP (*STandard for the Exchange of Product model data*) [137] est couramment utilisé pour échanger des modèles de pièces et de produits entre différents logiciels de Conception Assistée par Ordinateur (CAO).

A mesure que le nombre de protocoles, de standards et de solutions techniques pour assurer la communication entre deux systèmes augmente, la complexité pour intégrer de nouveaux systèmes augmente également considérablement. Lors de la mise en connexion de deux systèmes, l'environnement

2. `https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=mqtt`

hétérogène des entreprises nécessite toujours de combiner et d'exploiter divers protocoles et standards de communication avec de nombreux formats d'échanges. De cette façon, pour assurer l'interopérabilité des systèmes, chaque système doit être en mesure d'échanger des données à travers des connexions très spécifiques, qui emploient un protocole ou un standard de communication ainsi qu'un format d'échanges pour les données.

1.2.1.3 Hétérogénéité conceptuelle

La représentation conceptuelle d'une donnée permet d'exprimer sa sémantique, en d'autres termes, son sens du point de vue de la compréhension. Cependant, selon les domaines et les utilisateurs, une donnée peut avoir plusieurs représentations conceptuelles [35]. Gärdenfors et Zenker [194] rappellent que les conflits sémantiques sont provoqués par des perspectives multiples, à savoir des vues différentes de la donnée. Une opération de mappage, qui vise à établir des relations d'équivalence entre les différentes représentations conceptuelles d'une même donnée, est donc nécessaire [86]. Une donnée peut être modélisée et écrite de façon très variée, entraînant à la fois des conflits structuraux (données représentées par une structure et des concepts différents d'un modèle à l'autre) et sémantiques (données interprétées de façon différente selon le domaine). La prolifération des représentations conceptuelles des données complexifie inévitablement la réalisation des mappages.

Étant donné que chaque représentation conceptuelle, ou modèle, utilise sa propre structure et sa propre terminologie pour décrire ses données, des ambiguïtés de sens peuvent inévitablement intervenir et limiter la compréhension des messages échangés. Les conflits sémantiques sont considérés comme l'une des principales barrières qui empêchent l'intégration de nouveaux systèmes et, *in fine*, de déployer, à pleine capacité, un environnement autonome où l'échange et le partage des données est fluide et continu.

1.2.1.4 Hétérogénéité des données

Selon Chen *et al.* [38], les données industrielles peuvent être caractérisées par 5V : (1) volumineuses (une grande quantité de données) ; (2) variées (les données sont présentes sous différentes formes) ; (3) véloces (les données changent constamment) ; (4) versatiles (les données sont biaisées, incohérentes, approximatives et incomplètes) ; (5) valuables (les données, recels d'informations). L'hétérogénéité des données se traduit par différentes façons d'écrire une même donnée. Des conflits sémantiques peuvent intervenir lorsque les données n'utilisent pas la même granularité, le même langage, ou encore lors de l'emploi de synonymes. Des conflits syntaxiques peuvent également être introduis, lorsque des préfixes et des suffixes sont rajoutés aux données. Par ailleurs, Chen *et al.* insistent sur le fait qu'il est nécessaire de considérer que les données peuvent être erronées ou mal écrites.

1.2.1.5 Instablité de l'environnement et des systèmes

L'hétérogénéité des systèmes impose impérativement le développement d'adaptateurs *ad-hoc* spécifiques à chaque type de communication. Ces adapateurs sont généralement édités à la main pour

gommer les différences des systèmes et nécessitent un fort heuristique métier concernant les deux systèmes ou domaines à relier. Or, dans un environnement numérique industriel instable et dynamique, provoqué par l'émergence de nouvelles technologies, par la prolifération des nouvelles sources d'information et par la nécessité d'adapter, de reconfigurer et de faire évoluer la chaîne de valeur pour répondre aux fortes demandes de personnification des produits, ces adaptateurs statiques ne semblent pas être une solution optimale sur le long terme et requièrent un investissement considérable pour les maintenir en état de fonctionner dans un environnement en pleine mutation. Intégrer de nouvelles sources d'information est une activité complexe dans un environnement préexistant.

Les systèmes sont inévitablement dynamiques, dus aux reconfigurations, aux évolutions et aux changements de leurs spécifications [126]. L'instabilité de l'environnement, cumulée avec l'hétérogénéité des ses systèmes, complexifie le remaniement des connexions existantes. Maintenir l'interopérabilité des systèmes dans un environnement numérique en perpétuel changement et extension est une activité délicate et extrêmement coûteuse.

1.2.2 Analyse et discussion des grandes approches pour l'interopérabilité

Garantir l'interopérabilité des systèmes dans un environnement numérique instable et hautement hétérogène composé d'une multitude de standards, de protocoles de communication, de formats de données et de données hétéroclites isolées, constitue un défi d'envergure depuis maintenant plusieurs années. Les approches intégrées, unifiées et fédérées constituent les trois grandes approches permettant d'instaurer un climat d'interopérabilité entres les systèmes et les organisations. Cependant, ces trois approches répondent-elles aux exigences de l'industrie 4.0 ? Plus particulièrement, vis-à-vis des propriétés de l'environnement identifiées précédemment en Section 1.1, sont-elles en mesure de respecter les propriétés de diversité, d'autonomie, de dynamisme, et d'évolutivité et de connectivité des systèmes ?

Il est nécessaire de mesurer l'impact des approches pour l'interopérabilité, à savoir les architectures intégrées, unifiées et fédérées pour chacune des propriétés de l'environnement. Ces propriétés doivent être préservées pour garantir l'exploitation des technologies de l'Industrie 4.0 et promouvoir l'émergence de l'industrie intelligente. Étant donné que les propriétés d'interdépendance et d'émergence dépendent de la propriété qui permet la connexion entre les systèmes, ces deux propriétés n'ont pas été prises en considération lors de l'analyse. Le Tableau 1.3 résume les résultats de l'analyse.

L'impact de chacune des approches pour l'interopérabilité, vis-à-vis des propriétés de l'environnement de l'Industrie 4.0, a permis d'identifier l'approche qui semble la plus souhaitable pour répondre aux prérequis de l'industrie intelligente. Le Tableau 1.4 illustre, pour chaque propriété d'un système au sein d'un CPS, l'impact des approches intégrées, unifiées et fédérées. Ces propriétés doivent être au maximum préservées, de sorte à répondre aux enjeux de l'Industrie 4.0.

Architecture	*Dynamique*	*Évolutif*	*Autonome*	*Connecté*	*Diversifié*
Architecture intégrée	Le couplage étroit entre les systèmes impose une architecture fixe et rigide. L'architecture intégrée force les systèmes à adopter un standard et restreint l'ajout de nouveaux systèmes.	Le besoin d'un système change fréquemment, or, une architecture intégrée est extrêmement sensible aux modifications. L'évolution d'un système peut engendrer des perturbations disparates au sein de l'architecture.	Inévitablement, si les systèmes ne sont pas libres d'évoluer à leur convenance, alors cela se traduit par une perte de liberté et donc par un manque d'autonomie. L'autonomie des systèmes est inhibée par un formalisme contraignant.	Les relations entre les systèmes sont explicites et exprimées par le métamodèle partagé de l'architecture intégrée. Le couplage étroit permet l'échange de données sans restriction ni ambiguïté.	Migration d'un environnement hétérogène vers un environnement homogène où les modèles suivent les recommandations du métamodèle intégré pour satisfaire le besoin de communication et d'interaction des systèmes.
Architecture unifiée	Dépend en partie du maintien à jour du métamodèle unifié. L'ajout de tout nouveau système doit être reflété dans le métamodèle unifié.	Le métamodèle unifié impose son formalisme aux métamodèles locaux. L'évolution des métamodèles est contrainte de respecter le standard de référence, ou, doit être reflété au niveau du métamodèle unifié.	Les systèmes sont libres d'évoluer à leur convenance. Cependant, il faut garder à l'esprit que l'évolution locale d'un métamodèle n'est pas forcément répercutée au niveau du métamodèle unifié.	Le métamodèle unifié sert de cadre de compréhension commun à tous les systèmes. Des équivalences sémantiques entre les concepts des métamodèles peuvent être établies. Le métamodèle unifié ne peut pas détenir l'intégralité de la sémantique du domaine.	La communication entre deux systèmes se fait d'un commun accord par l'intermédiaire d'un métamodèle de référence. Cependant le métamodèle ne garantit pas la diversité en cas d'évolution des systèmes.
Architecture fédérée	L'architecture est ouverte et permet d'ajouter ou de retirer des systèmes. Aucun standard n'est prédéfini et aucun standard ne bride l'accès à la fédération.	L'approche fédérée minimise la perte de liberté des systèmes. Ils sont donc libres d'évoluer à leur convenance.	L'individualité, l'autonomie et l'indépendance des systèmes sont préservées. Les systèmes repondent à leurs propres règles et communiquent à travers leurs interfaces.	La connexion à travers les interfaces des systèmes est spécifique à chaque communication. L'ouverture d'un système n'est pas toujours garantie et complique naturellement l'interconnexion des systèmes.	L'architecture fédérée promeut un environnement numérique hautement hétérogène.

TABLE 1.3 – Analyse des approches intégrées, unifiées et fédérées vis-à-vis des propriétés de l'environnement de l'Industrie 4.0

Approches	*Solutions*	*Dynamique*	*Évolutif*	*Autonome*	*Connecté*	*Diversifié*
Intégrée	Plateformes *cloud*	✗	✗	✗	✓	✗
	Architectures référence	✗	✗	✗	✓	✗
Unifiée	Médiateur	✓	✓	✓	✓	✓
Fédérée	Multiples Métamodèles Ontologies	✓	✓	✓	✓	✓

TABLE 1.4 – Impacte des approches intégrée, unifiée et fédérée vis-à-vis des propriétés des systèmes au sein d'un CPS

Approche intégrée. Les approches intégrées ne traitent pas directement les problèmes d'interopérabilité entre les systèmes hétérogènes. Les architectures intégrées procurent un cadre d'interopérabilité limité à un contexte bien précis, qui n'a pas pour objectif de tacler dans sa globalité les problèmes liés à l'hétérogénéité des systèmes. L'adaptation des systèmes aux spécifications de l'architecture intégrée permet la migration d'un environnement hétérogène vers un environnement homogène.

Les notions d'interopérabilité et d'intégration sont deux phénomènes différents. Contrairement à la notion d'intégration, dans laquelle les systèmes sont étroitement couplés par des mécanismes de connexions rigides, la notion d'interopérabilité, quant à elle, met en avant un couplage des systèmes lâches, à savoir un mécanisme de connexion flexible, apte à appréhender les environnements dynamiques[36, 151]. Le couplage lâche des systèmes, à travers des interfaces prédéfinies et disponibles, garantit leur autonomie et leur indépendance [180], ce qui n'est pas le cas lors d'un couplage étroit entre les systèmes. Un changement local dans l'architecture intégrée peut impliquer des changements plus généraux, qui ne sont pas forcément souhaités.

Approche unifiée. Dans la mesure où plusieurs systèmes doivent interopérer, chacun d'entre eux est sensé préserver son autonomie de fonctionnement. Or dans le cas des architectures unifiées, l'interopérabilité entre les systèmes est guidée par une représentation conceptuelle partagée par tous, i.e. un métamodèle pivot ou métamodèle unifié. Partager un vocabulaire commun de compréhension signifie en contrepartie contraindre plusieurs systèmes à utiliser un unique champ lexical, et par conséquent, assumer que des pertes d'informations lors de la traduction soient possibles en cas d'évolution des métamodèles locaux des systèmes [64]. Le processus de traduction est sensé déterminer les représentations équivalentes (entre une ontologie source et une ontologie cible par exemple) pour une même donnée et procéder à l'alignement de ces deux représentations. Or, la traduction échoue si deux concepts des métamodèles locaux différents ne sont pas sémantiquement liés par l'intermédiaire d'un métamodèle pivot. Il est légitime de se demander si le métamodèle unifié est en capacité d'évoluer à la demande et d'intégrer efficacement les évolutions des métamodèles locaux. L'utilisation d'un métamodèle pivot

nécessite inévitablement un mécanisme de mise à jour périodique, pour absorber les évolutions au niveau local [172].

Dans le cadre d'une médiation sémantique, une ontologie de référence, ou un métamodèle unificateur peuvent jouer le rôle de métamodèle pivot. Concernant les ontologies de référence, une et une seule ontologie médiatrice est utilisée pour créer la jonction entre l'ensemble des métamodèles locaux [43]. Cependant, l'ensemble des conflits sémantiques ne sont pas intégralement résolus. Par défaut, les ontologies ne représentent qu'une part très spécifique de la connaissance du domaine dans lequel elles interviennent. Par conséquent, elles ne couvrent pas l'intégralité d'un domaine [134]. Il existera toujours des systèmes avec des métamodèles locaux non-conformes au métamodèle pivot à intégrer à l'architecture.

La prolifération des ontologies dans un même domaine spécifique a pour conséquence d'encourager le silotage de l'information. Notons également que chaque ontologie dispose de son propre formalisme (structure du modèle) et de son propre langage (terminologie des concepts). Gyrard *et al.* [72] soutiennent les pratiques qui visent à réutiliser et à aligner les ontologies existantes, pour réduire l'hétérogénéité structurelle et terminologique entre les ontologies, de sorte à accroître l'interopérabilité sémantique entre les systèmes. Gyrard et Serrano [71] soulignent l'importance d'unifier les ontologies et leur vocabulaire, pour uniformiser l'annotation sémantique des données. Cependant, les conflits sémantiques au niveau conceptuel sont la conséquence directe de la multiplication d'ontologies hétérogènes. Pour pallier à ces problèmes, de nombreux travaux proposent d'identifier des relations de correspondance, i.e. des relations sémantiques inter-ontologies, pour aligner et fusionner les éléments (classes, relations, attributs) d'ontologies différentes [116]. L'article présenté par Shvaiko et Euzenat [150] reprend les dernières avancées (avant 2013) dans le domaine d'étude de l'alignement des ontologies. Ainsi, lorsqu'il s'agit d'utiliser un métamodèle pivot, les problèmes liés à l'hétérogénéité des données migrent au niveau conceptuel.

Les approches visant à aligner et fusionner des ontologies médiatrices sont similaires à celles qui tendent à aligner les concepts de deux métamodèles locaux. De plus, notons que l'utilisation d'un métamodèle pivot est un moyen déporté, i.e. un moyen intermédiaire, pour promouvoir un cadre de compréhension commun à tous les systèmes. Il est alors légitime de se demander si ce n'est pas plus efficace de tisser directement, sans intermédiaire, un mappage sémantique entre les métamodèles de chaque système ?

Ne perdons pas de vue que l'hétérogénéité sémantique est avant tout un problème provoqué par des perspectives multiples, i.e. des points de vue différents de la donnée [194]. Le concept d'un domaine spécifique peut être sans ambiguïté dans un certain contexte, tandis que le même concept sera ambigu dans un autre type de contexte. Par conséquent, les conflits sémantiques ne peuvent être réduits par la simple labellisation d'un concept, i.e. l'établissement d'un terme représentant le sens d'un concept. La labellisation d'un concept dépend avant tout de la perspective dans lequel le concept est employé. Ainsi, un concept pourrait/devrait détenir plusieurs labels. Or, une ontologie de référence tente d'harmoniser et d'agréger plusieurs représentations sous un unique label.

Au final, le cœur du problème d'interopérabilité des systèmes au niveau sémantique est étroitement corrélé au problème communément appelé *Symbol Grounding*, qui cherche à représenter fidèlement le

sens concret d'un concept (d'un symbole), quel que soient ses perspectives [77, 42]. Dans le cadre de l'approche unifiée, il semble alors vraiment compliqué d'établir un unique concept pivot, auquel peut se rattacher les concepts de plusieurs systèmes indépendants.

Approche fédérée. Par nature, l'interopérabilité est un phénomène fédéraliste qui réduit au minimum le manque de liberté des systèmes et augmente leur acceptance à s'interconnecter [151]. Un système libre se traduit par un niveau d'autonomie et d'indépendance bénéfique à son évolution individuelle, tandis que la capacité d'un système à interopérer, via ses interfaces avec d'autres systèmes, garantit un climat de coopération et d'interdépendance.

Cependant, il est important de garder à l'esprit que l'interconnexion des systèmes nécessite que leurs interfaces soient disponibles, ouvertes [133].

1.2.3 Problématique

Les approches d'interopérabilité basées sur une technologie ou un standard, ne semblent pas alignées avec les exigences d'une interopérabilité qui est sensée promouvoir l'interconnexion des systèmes dans un environnement dynamique et évolutif [5]. La prolifération d'un écosystème de standards et de technologies ne résout que localement, et pour des environnements très spécifiques, les problèmes d'interopérabilité. Les approches basées sur un standard force les systèmes à suivre une norme unificatrice, or, il existera toujours des systèmes non conformes à interconnecter [142]. Les standards sémantiques, tels que les métamodèles pivots (ontologies de références, ontologies médiatrices, métamodèles unificateurs...), ne procurent pas un cadre propice à une interopérabilité dynamique et évolutive. Par défaut, un standard sémantique restreint les systèmes à se conformer à un unique cadre de compréhension [108].

Une architecture, qui supporte le couplage lâche entre les systèmes, est nécessaire pour satisfaire le besoin d'évolutivité et d'adaptabilité des systèmes dans un environnement hautement dynamique [132]. L'approche fédérée semble être la clé pour aboutir à une interopérabilité sans restriction, qui préserve les propriétés des systèmes et de l'Industrie 4.0. Une architecture fédérée promeut un couplage lâche entre les systèmes, qui permet, d'une part, d'améliorer la modularité de l'architecture, d'autre part, de faciliter l'ajout et le retrait de systèmes.

Considérant les défis pour l'interopérabilité dans un contexte d'Indutrie 4.0 (voir Section 1.1.4.3), à savoir la création d'un mécanisme de connexion générique, automatique et dynamique, une problématique de recherche a été définie. La problématique repose également sur l'analyse des grandes approches pour l'interpérabilité par rapport aux propriétés de diversité, d'autonomie, de dynamisme, d'évolutivité et de connectivité des systèmes. La convergence de ces deux analyses soulève de nombreuses questions.

Pour commencer, l'approche fédérée permet un couplage lâche entre les systèmes, par l'intermédiaire de leurs interfaces. Or, les interfaces de chacun des systèmes sont toutes différentes. De la même façon, chaque système détient son propre modèle de données avec sa propre sémantique. Lorsqu'il s'agit de coupler, i.e. d'interconnecter deux systèmes, la connexion établie entre eux sera unique, spécifique à la communication entre ces deux systèmes. **Il est alors nécessaire de se demander si, dans**

le contexte d'une fédération, le mécanisme de connexion peut satisfaire les besoins de généricité et d'automatisation de l'interopérabilité ? Plus précisément, est-il possible de concevoir un mécanisme de connexion générique, automatique et dynamique dans le cadre d'une architecture fédérée ? Étant donné que l'approche fédérée prône une grande liberté d'évolution de ses systèmes, est-il réellement possible d'automatiser la mise à jour du connecteur, de sorte que les nouvelles spécifications soient prises en considération ?

Dans ce manuscrit de thèse, une réponse concrète sera apportée à chacune de ces questions.

1.3 Proposition

L'un des enjeux majeurs de l'interopérabilité est le développement de mécanismes de connexion durable, capables de s'adapter automatiquement aux changements de spécifications des systèmes à venir, et ce, avec un minium d'effort [4]. Dans un environnement en pleine mutation et en perpétuelle évolution, les mécanismes de connexion doivent détenir la capacité de s'auto-ajuster dynamiquement, pour absorber les modifications des systèmes [139]. Panetto *et al.* décrivent la *Prochaine Génération de Système d'Information d'Entreprise* comme un système fédérateur dirigé par les modèles, où l'interconnexion des systèmes est réalisée par l'intermédiaire de leur représentation conceptuelle [133]. De ce fait, la modularité de l'architecture fédérée se traduit par une composition, ou assemblage de plusieurs modèles capables de s'auto-adapter et de s'auto-ajuster individuellement et collectivement.

Une architecture modulaire et dynamique nécessite un mécanisme d'intéroperabilité *plug and play* [48, 87]. En principe, une approche *plug and play* permet automatiquement de connecter deux ou plusieurs systèmes, tout en minimisant l'intervention humaine. Ce type de mécanisme de connexion garantit une meilleure flexibilité et une meilleure reconfigurabilité des chaînes de production, en garantissant la construction de SoS spécifiques à un besoin [136]. De ce fait, pour répondre à un besoin particulier, une composition de plusieurs systèmes sera spécialement construite grâce à l'interconnexion de leur modèles.

1.3.1 Spécifications de l'interopérabilité *plug and play* pour la continuité numérique

Il est important de comprendre les technologies à mettre en oeuvre pour atteindre les spécifications d'une interopérabilité *plug and play* qui facilitera, à terme, la création de chaînes de production efficientes et reconfigurables selon les besoins, permettant la convergence des données et la continuité numérique.

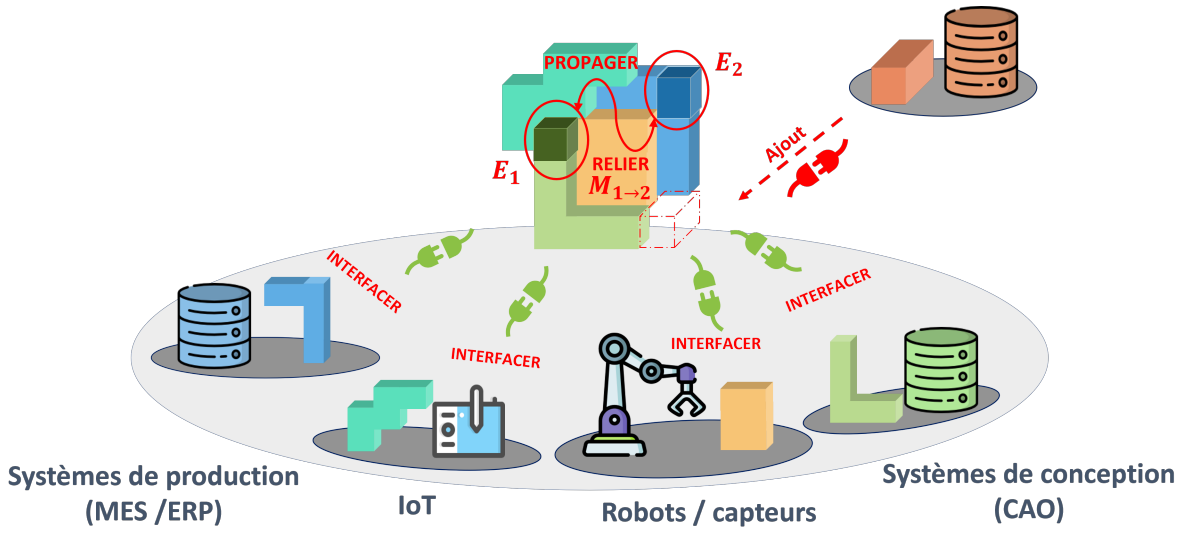


FIGURE 1.5 – Architecture *plug and play* Dirigée par les Modèles

Le concept présenté suit les principes dictés par le domaine de l'Architecture Dirigée par les Modèles (*Model Driven Architecture* MDA). De cette façon, chaque système est représenté par un métamodèle ayant une structure et une terminologie propre. Des fonctions, représentées dans la Figure 1.5, ont été imaginées pour implémenter la vision d'une interopérabilité *plug and play* entre les systèmes :

– *Fonction interfacer.* Dans un environnement numérique qui mute sans cesse, l'ajout, le retrait, ou encore la modification des systèmes sont autant de scénarios qui ne doivent pas compromettre le fonctionnement global de la fédération. Cette fonction concerne l'habilité à connecter n'importe quel système, qu'il soit physique ou numérique, quelle que soit la technologie employée et la représentation structurelle, syntaxique et sémantique de la donnée. Cette fonction permet d'afficher et de reconnaître les interfaces (physiques, protocole de communication, structure, syntaxe et sémantique des données) d'un système vers un autre.

  Exemple : un capteur de température affiche sa connectique USB dans un format 8bit pour diffuser la température. Un modeleur CAO affiche son modèle de données (complétement ou partiellement) via des API structurelles et sémantiques reconnues par les autres systèmes.

– *Fonction relier.* Dans un environnement numérique collaboratif, le partage et la combinaison de connaissances entre les systèmes sont fondamentaux pour faciliter leurs interactions via les interfaces. La notion de mise en relation (i.e. *mapping*) consiste à créer des liens, des relations structurelles et sémantiques entre les concepts de plusieurs modèles, pour assurer un cadre de compréhension et de cohérence pour les données échangées [173]. Selon Panetto *et al.* [133], la fonction de mise en relation des modèles de données réfère à la capacité des systèmes à tisser des connexions ambiantes entre "*everything, anywhere and anytime*" indépendamment de l'hétérogénéité des modèles. Cette mise en relation peut être "faible" (on sait qu'il existe une relation entre deux modèles qui n'est pas formalisée) ou "forte", ce qui permettrait de propager les évolutions d'un modèle vers les autres.

Exemple : la même information est représentée par des concepts différents, i.e. par une structure, une syntaxe ou une sémantique différente d'un modèle à l'autre. Le mappage M_1 relie l'élément E_1 du modèle vert partagé à l'élément E_2 du modèle bleu partagé.

– *Fonction propager.* Constater et propager l'impact du changement d'une donnée sur une autre est l'essence même de la continuité numérique. Cette dernière notion fait directement écho à la propagation des changements inter modèles [100]. Les mécanismes de changement peuvent être "faibles" (simple alerte) ou "forts" (évolution formelle des modèles), en adéquation avec le niveau de mise en relation.

Exemple (changement inter modèles) : la modification de l'élément E_1 du modèle vert partagé entraîne le changement d'état de l'élément E_2 du modèle bleu partagé.

Pour résumer, un mécanisme d'interoperabilité *plug and play*, pour la continuité numérique, repose sur la capacité (1) d'interfacer les modèles de données les uns avec les autres ; (2) de mettre en relation les concepts des différents modèles ; (3) de propager les changements de données inter-modèles. La solution proposée reprend les principes de l'approche fédérée : il n'y a pas de format (structure, syntaxe et sémantique) de données commun et aucun système n'impose le formalise de son modèle de données. Par exemple, selon la Figure 1.5, le modèle rouge représente un nouveau système à intégrer à notre fédération ; (1) la structure, la syntaxe et la sémantique du modèle rouge sont reconnues ou apprises par le système ; (2) des liens de mise en relation sont tissés entre les concepts du modèle rouge et les concepts des modèles connus par le système ; (3) une fois les données entre modèles reliées, la propagation des données peut avoir lieu, et par extension la continuité numérique sera assurée.

1.3.2 Une Interoperabilité Dirigée par les Modèles

L'ingénierie dirigée par les modèles (*Model-Driven Engineering* MDE) est une approche qui provient de la communauté de l'ingénierie logicielle et qui favorise l'utilisation de "modèles" comme concept unificateur pour représenter un objet, un système ou encore une partie d'un système [18]. Les informations d'un système sont alors encapsulées dans un modèle spécifique. Selon les principes dictés par l'approche MDE, un modèle (M1) est une représentation abstraite de la réalité. Le métamodèle (M2), ou langage de modélisation, est quant à lui le langage qui décrit comment un objet du monde réel est modélisé d'un point de vue structurel et terminologique. Par conséquent, le modèle est dit conforme à son métamodèle (exemple : une carte (M1) est conforme à sa légende (M2), la carte seule ne peut être interprétée sans sa légende). Les modèles sont généralement décrits selon les spécifications du langage unifié UML (*Unified Modeling Language*), qui procure un cadre de modélisation commun. Un système complexe peut être décomposé en plusieurs métamodèles, qui procurent une abstraction pour chacune des préoccupations désirées. En d'autres termes, un système peut être représenté sous différents points de vue pour réduire sa complexité [146]. Dans le cas d'un SoS, et plus particulièrement d'un CPS, chacun des systèmes est formellement modélisé [166].

> *Il y a deux principes fondamentaux qui imposent la décomposition d'un système en sous-parties plus petites. D'une part, les systèmes modernes sont devenus trop complexes pour qu'on puisse les appréhender dans leur totalité. D'autre part, il y a autant de niveaux de lecture que de catégories de lecteurs d'un modèle.*
>
> Thomas Bailet, Architecture logicielle, Edition ENI, 2016

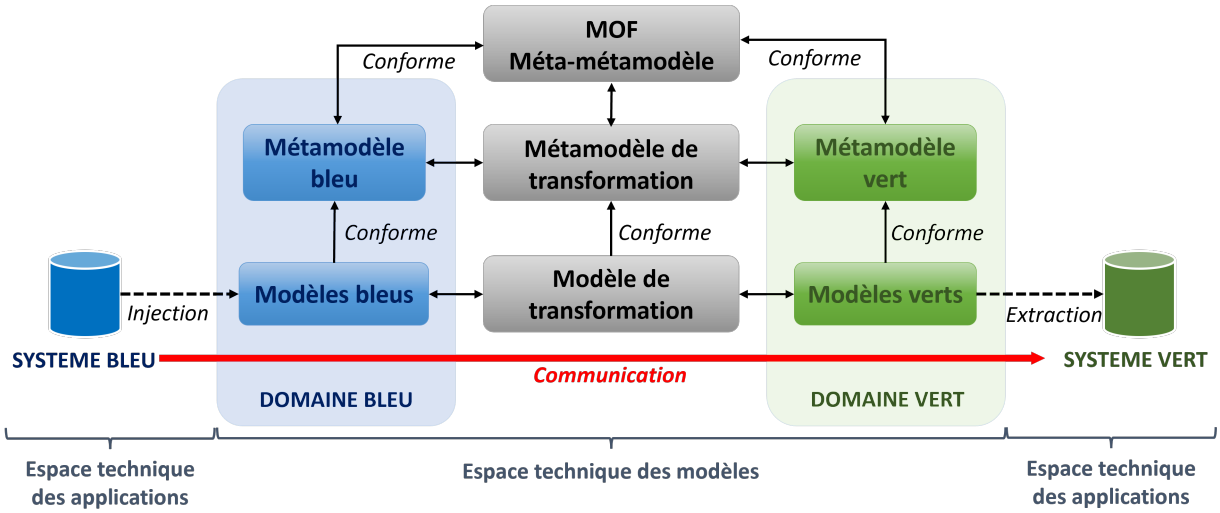


FIGURE 1.6 – Architecture Dirigée par les Modèles

Cependant, l'utilisation de modèles ne trouve pas uniquement son utilité dans une représentation structurée des données, mais également dans sa capacité à être transformée dans différents domaines sémantiques (i.e. différentes métamodélisations), ou entre différents niveaux d'abstraction, pour combler le fossé entre la conception et l'implémentassions dans une solution technique. En accord avec l'Architecture Dirigée par les Modèles (*Model Driven Architecture* MDA) [152, 96], un tel mécanisme de traduction *Model-to-Model* (M2M) permet de spécifier les relations de correspondance entre différents métamodèles. Les techniques de transformation de modèle [148, 19] constituent la pierre angulaire de l'approche MDA. Elles permettent de transformer un modèle source dans un contexte spécifique en un modèle cible dans un autre contexte particulier. Autrement dit, la transformation qui lie les modèles source et cible se caractérise par un ensemble de règles qui relient les concepts du métamodèle source aux concepts du métamodèle cible. Ces règles de transformation définissent les relations structurelles (relation entre méta-concepts ayant des structures différentes) et sémantiques (relation entre méta-concepts ayant la même signification, i.e. le même sens) qui relient deux métamodèles. Les règles de transformation jouent le rôle de fonction d'interopérabilité sémantique et structurelle entre différents modèles, par l'intermédiaire de leur métamodèles [192].

Suivant la Figure 1.6, les systèmes bleu et vert sont respectivement méta-modélisés par les méta-modèles bleu et vert, qui procurent la sémantique et la structure des données des deux systèmes. Le système bleu initie une communication avec le système vert par l'intermédiaire de *l'espace technique*

des modèles. Selon l'architecture MDA [84], les données du système bleu sont alors injectées au sein d'un modèle bleu qui sera ensuite transformé suivant les règles de transformation définies par le modèle de transformation. La transformation du modèle bleu aboutira à la création d'un modèle vert conforme à la sémantique et à la structure du métamodèle vert. Les données du modèle vert seront ensuite extraites puis intégrées au sein du système vert. Ainsi, le modèle de transformation facilite la communication entre les systèmes et joue le rôle clé de la fonction d'interopérabilité, en traduisant les données échangées du domaine bleu au domaine vert. MDA a été développé pour surmonter la grande diversité des systèmes, y compris l'hétérogénéité de leurs plates-formes (langages de programmation utilisés) et l'hétérogénéité de leurs supports de données (bases de données). L'accent est donc mis uniquement sur l'interopérabilité des modèles, et plus particulièrement sur l'interopérabilité sémantique, en établissant des relations structurelles et sémantiques. Notons qu'un troisième niveau (M3) est présent sur la figure. Une architecture MDA propose un troisième niveau qui permet la création de différents métamodèles (M2) conformes aux spécifications d'un méta-métamodèle (M3) unificateur. Ainsi, selon le méta-métamodèle, tout métamodèle est constitué de classes, d'associations qui relient les classes et de propriétés qui composent la classe. Le méta-métamodèle MOF [3] (*Meta-Object Facility*) procure un cadre de compréhension commun à tous les métamodèles. Ce cadre partagé par tous les métamodèles permet la transformation de modèles.

Les techniques de transformation de modèles constituent les fondations d'une intéropérabilité dirigée par les modèles (*Model Driven Interoperability*) [24]. Deux types de transformations existent :

- *Transformation verticale.* Permet de passer d'un modèle indépendant de toute plateforme (*Platform Independent Model* PIM), à savoir indépendant de toute spécification technique, à un modèle spécifique à une plateforme (*Platform Specific Model* PSM). En général, les transformations verticales permettent de relier différents niveaux d'abstraction. Cette pratique simplifie l'implémantation d'un modèle dans une syntaxe propre à une solution technique.
- *Transformation horizontale.* Permet la conversion d'un modèle PSM vers un autre modèle PSM, dans une syntaxe et une terminologie totalement différentes. Généralement, les transformations horizontales opèrent lorsqu'il s'agit de passer d'un domaine sémantique à un autre.

L'interopérabilité dirigée par les modèles apparaît aujourd'hui comme une solution concrète pour simplifier la création de mécanismes de connexions entre différents modèles, et plus particulièrement, entre plusieurs systèmes. Elle permet l'accès à la vision d'une industrie modulaire, où la composition de systèmes et de services permet de répondre, avec flexibilité, à un besoin spécifique [3, 67].

1.3.3 Proposition fondamentale

L'interopérabilité des modèles, que prône l'approche MDI, repose principalement sur la capacité de mapper les concepts de différents métamodèles, pour ensuite décrire les règles de transformation qui les relient [191]. Or, définir des relations entre des métamodèles hétérogènes, possédant leurs propres structures et terminologie, est une activité très délicate, qui demande du temps et du savoir-faire. Jusqu'à présent, les approches permettant la transformation automatique des modèles s'appuyaient sur la

3. `https://www.omg.org/mof/`

définition de langage de transformation qui permettaient de spécifier, à la main, les règles de transformation entre deux métamodèles. Les règles de transformation éditées entre deux métamodèles (M2) agissent sur les éléments des modèles (M1). Par nature, ces règles sont dites descriptives puisqu'elles sont spécifiques à un modèle de transformation qui relie deux ou plusieurs métamodèles distincts.

Un grand nombre de langages de transformation et de plateformes ont vu le jour ces dernières années pour simplifier l'édition de transformation de modèles. Dans leur article, Kahani *et al.* [89] proposent une classification de ces différents langages et outils. Parmi les langages de transformation, l'OMG fournit le standard QVT (*Query View Transfromation*) [130], qui procure une architecture et des langages dédiés facilitant la transformation de modèles. Le langage ATL (*ATLAS Transformation Language*) [19], inspiré du standard QVT, fournit un langage et une syntaxe spécifique, permettant l'édition de règles de transformation. Des outils orientés transformation de graphes sont proposés par les plateformes VIATRA [44] et AToM [103].

Cependant, écrire à la main des règles de transformation *ad hoc*, tout en respectant la syntaxe d'un langage de transformation, ne répond pas aux défis de généricité, d'automatisme et de dynamisme d'une interopérabilité en phase avec les exigences de l'Industrie 4.0. Rédiger des règles de transformation nécessite une excellente compréhension de la sémantique des deux domaines à relier, ainsi qu'une connaissance des langages de transformation pour formaliser le mappage. En outre, un effort important est nécessaire pour décrire et maintenir les modèles de transformation dans un environnement industriel très hétérogène, qui ne cesse de croître et de se complexifier. Des mécanismes de réutilisation des modèles de transformation ont été proposés, pour éviter d'avoir à mettre en œuvre une nouvelle transformation à partir de zéro, à chaque fois qu'un nouveau contexte et que de nouveaux besoins apparaissent [99, 30]. Cependant, la nécessité de construire un modèle de transformation à partir de zéro reste une situation inévitable lorsque l'on considère une approche *plug and play* pour l'interopérabilité des systèmes.

L'automatisation des transformations de modèles semble pour le moins compliquée. Selon Nilsson *et al.* [126], un mécanisme de connexion évolutif et dynamique nécessite la création automatique de relations sémantiques entre les différents modèles. En d'autres termes, la création d'un mécanisme de connexion automatique de type *plug and play* doit être en mesure d'interpréter et de traduire les modèles dans différents formats [122]. La notion d'interprétation fait directement écho à la nécessité de comprendre, et donc, d'intelligence. Lee *et al.* [105] parlent par exemple de connecteurs intelligents. Zhong *et al.* [198] introduisent la notion d'interconnexion intelligente entre les systèmes. Lelli *et al.* [107] utilisent le concept d'interopérabilité intelligente pour introduire l'importance de l'interconnexion des systèmes dans la vision de l'Industrie intelligente.

> *Discovering semantic information and resolving mismatches requires the application of human intelligence and jugment [78]*

Partant du constat que le mécanisme de connexion imaginé par l'Industrie 4.0 doit comporter un certain degré d'intelligence pour automatiquement tisser des relations sémantiques entre différents modèles, une proposition fondamentale peut alors être définie.

Considérant les récentes avancées dans les domaines de l'apprentissage machine et de l'apprentissage profond, nous avons le sentiment, ou du moins l'intuition, que ces dernières prouesses peuvent simplifier et optimiser la création des modèles de transformation. Une série d'hypothèses de recherche peut alors être dressée :

(H1) La création du mécanisme de connexion permet une interopérabilité automatique entre les systèmes, par l'intermédiaire de leurs métamodèles, et ce, en minimisant les pertes sémantiques.

(H2) Le mécanisme de connexion répond aux spécifications d'interopérabilité *plug and play* entre les systèmes. Cela signifie que le dynamisme de l'environnement numérique et l'évolutivité des systèmes ne doivent en aucun cas être contraignants. De plus, les efforts humains pour créer le connecteur en termes de pré-traitement et post-traitement doivent être minimisés, si ce n'est être nul.

1.3.4 Objet d'étude

Maintenant que la proposition fondamentale est clairement exposée, un objet d'étude peut être à présent défini. **Étant donné que la proposition fondamentale porte sur un mécanisme de connexion, alors l'objet d'étude portera sur les modèles de transformation qui permettent à la fois les transformations verticales et horizontales des modèles.**

1.4 Synthèse

Dans ce premier chapitre, les propriétés de l'environnement de l'Industrie 4.0 sont explorées. Selon la vision de la 4$^{\text{ème}}$ révolution industrielle, l'industrie est un environnement instable, dynamique et hautement hétérogène. Les systèmes qui le compose doivent respecter des propriétés de dynamisme, d'évolutivité, d'autonomie, de connexion et de diversité.

Pour répondre aux contraintes de l'environnement, l'interopérabilité des systèmes doit répondre à trois grands défis : la création d'un mécanisme de connexion générique, automatique et dynamique.

Une proposition fondamentale appliquée à un objet d'étude a été élaborée vis-à-vis des défis et des besoins d'interoéprabilité de l'environnement de l'Industrie 4.0. Ces deux derniers éléments sont représentés dans la Figure 1.7. Comme expliqué dans l'introduction, une instanciation du métamodèle de la Figure 1 sera complétée au fur et à mesure de la lecture de ce manuscrit.

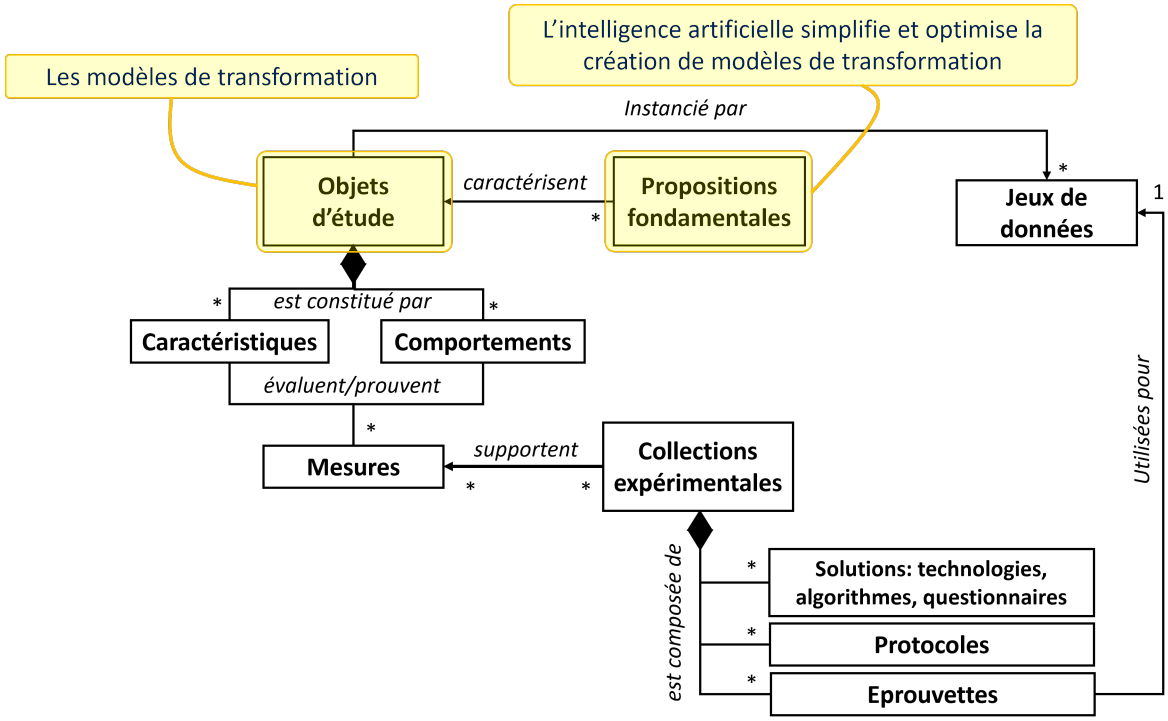


FIGURE 1.7 – Synthèse du Chapitre 1

Chapitre 2

Etat de l'art

Contenu

Maintenant que des problématiques ont été soulevées, et qu'une proposition fondamentale a été établie, une revue de l'état de l'art est nécessaire pour déterminer les bienfaits et les limites des approches existantes vis-à-vis de l'objet d'étude défini précédemment. L'analyse de l'état de l'art permettra la construction de la collection expérimentale, à savoir, un registre d'approches permettant de répondre totalement ou partiellement à la proposition fondamentale. Ces approches seront ensuite utilisées dans le cadre d'une étude comparative, pour mesurer leur pertinence par rapport aux propriétés de généricité, d'automatisme et de dynamisme des mécanismes de connexion à concevoir dans le cadre du déploiement d'une industrie 4.0.

Collection expérimentale

2.1 État de l'art de la fonction "relier"

Lors du Chapitre 1, trois fonctions ont été définies pour construire un mécanisme de connexion *plug and play* qui assure une interopérabilité dynamique entre les systèmes. Nous posons l'hypothèse que les interfaces des systèmes sont connues. Étant donné que la fonction "propager" repose au préalable sur le mappage, i.e. les relations établies par la fonction "lier", l'état de l'art présenté dans ce chapitre portera sur cette dernière fonction, qui, selon l'approche MDA, correspond aux transformations de modèles.

Selon Alwan *et al.* [8], les approches visant à tisser des relations entre deux représentations conceptuelles différentes des données peuvent être classifiées en trois catégories :

- *Approches basées sur les schémas.* La création des relations structurelles et sémantiques repose sur l'analyse des informations du schéma, à savoir la terminologie utilisée par les concepts (nom des classes, des attributs et des relations), ainsi que la structure interne et externe de ses concepts (agencement des classes les unes par rapport aux autres et attributs des classes).
- *Approches basées sur les instances.* La création du mappage est basée sur l'analyse des valeurs des attributs. De ce fait, il semble parfois plus simple de relier des attributs entre eux, en analysant auparavant leurs valeurs.
- *Approches hybrides.* Les approches hybrides tirent profit à la fois des avantages des approches basées sur les schémas et des approches basées sur les instances. La combinaison des deux précédentes approches permet parfois de combler les lacunes de l'une par les avantages de l'autre.

Selon le paradigme de l'ingénierie dirigée par les modèles, la notion de schéma est équivalente à la notion de métamodèle, tandis que la notion d'instance correspond à la notion de modèle. Cependant, il est important de noter et de souligner qu'il existe différentes façons, autres que les métamodèles proposés par l'approche MDA, pour modéliser et structurer la connaissance. Les graphes de connaissances et les ontologies sont des modélisations particulières de la connaissance.

A l'image des métamodèles, et comme toutes représentations conceptuelles, les graphes de connaissances et les ontologies sont des façons particulières de modéliser un domaine d'intérêt, en capturant

les concepts, les relations et les propriétés qui le définissent. Cependant ces représentations sont développées séparément et fonctionnent de manière isolées dans un contexte bien précis. Leur structure (ou typologie), leur terminologie, ainsi que la syntaxe de leurs données diffèrent. Par exemple, deux graphes de connaissances peuvent représenter un même domaine, sans pour autant utiliser la même structure et la même terminologie pour définir leurs concepts. D'ailleurs, il n'est pas rare que les graphes de connaissances soient complémentaires. Il en va de même pour les métamodèles et les ontologies qui proposent une vision restreinte et spécialisée d'un domaine. Pour intégrer ou fusionner efficacement différentes sources de données complémentaires, ayant des représentations conceptuelles différentes, il est nécessaire d'identifier et de mapper les concepts qui partagent le même sens, i.e. une sémantique commune.

Cette activité fait allusion au problème d'appariement (ou *matching*) et d'alignement des concepts issus de représentations conceptuelles totalement distinctes. L'objectif de ces deux méthodes est d'obtenir un mappage qui relie les représentations conceptuelles de deux sources de données différentes. Cependant, ne perdons pas de vue que l'opération de mappage n'est qu'une étape intermédiaire qui permet ensuite, à partir du mappage obtenu, de dériver les règles de transformation qui relient deux méta-modélisations différentes. Rappelons que les objectifs décrits dans ce manuscrit, tendent à développer un mécanisme générique, dynamique et automatique pour assurer la transformation de modèles. La Figure 2.1 illustre le processus de dérivation des règles de transformation dans sa globalité. L'état de l'art de la fonction "relier" sera donc divisé en deux axes : le premier concernera les opérations de mappage (voir Section 2.1.1), tandis que le second portera son attention sur les opérations de dérivation des règles de transformation à partir du mappage identifié au préalable (voir Section 2.1.2).

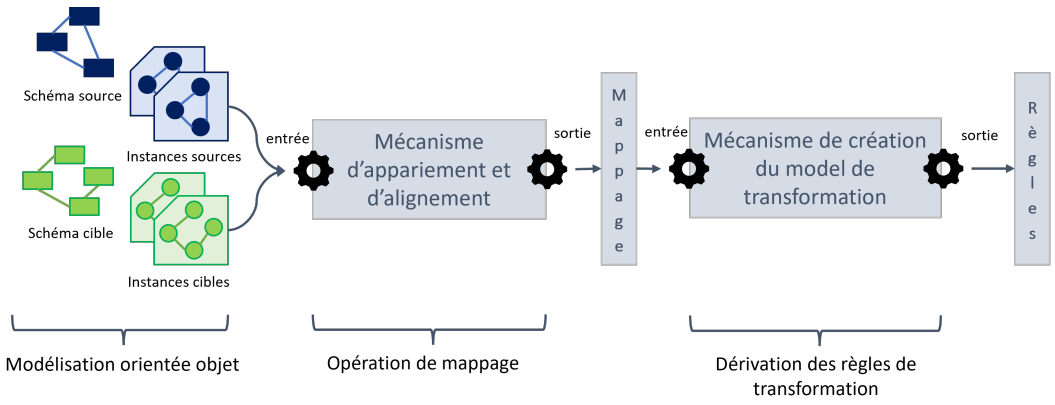


FIGURE 2.1 – Processus de dérivation des règles de transformation

2.1.1 Opération de mappage

Dans ce premier axe, une attention particulière sera portée sur les approches d'appariement et d'alignement de graphes et d'ontologies permettant d'obtenir un mappage entre différentes représentations conceptuelles. Les opérations d'appariement et d'alignement consiste à identifier les concepts, provenant de différentes représentations conceptuelles, qui partagent un sens commun et qui se réfèrent

au même objet dans le monde réel. De cette façon, le mappage M obtenu peut être défini de la façon suivante :

$$M = \{(c_s, c_c) \in R_s \times R_c | c_s \equiv c_c\}$$

où c_s et c_c correspondent à des concepts équivalents, provenant respectivement des représentations conceptuelles source R_s et cible R_c.

2.1.1.1 Opération de mappage entre graphes

Par définition, un graphe de connaissances (*Knowlege Graph KG*) est constitué d'un ensemble de noeuds (entités E), reliés par des arcs (relations R). Ces entités sont composées de propriétés (attributs A) qui encapsulent des valeurs V. Un graphe de connaissances peut alors être représenté par l'équation $G = (E, R, A, V)$. Les graphes sont largement utilisés pour capitaliser et représenter la connaissance, ainsi que les relations sémantiques qui relient deux entités. Parmi les bases de données orientées graphes les plus connues, *DBpedia* [106] recense, par exemple, les connaissances provenant de *Wikipedia*. Les mécanismes d'appariement et d'alignement jouent un rôle important dans la construction et la fusion de KG hétérogènes. Intégrer plusieurs graphes au sein d'un seul et unique KG plus grand, permet d'appliquer des mécanismes de raisonnement qui s'appuient sur des données cohérentes et complètes. Les mécanismes de raisonnement appliqués aux KG permettent d'initier de nombreuses applications, telles que, les systèmes de recommandation, de réponse aux questions ou encore de recherches sémantiques [39].

L'apprentissage de représentation des KG semble être une solution prometteuse pour édifier le mappage qui relie deux entités provenant de deux KG différents. L'idée principale consiste à projeter les entités de chaque KG dans un espace vectoriel, en encodant leur représentation (*node embedding*), puis d'apprendre une fonction de similarité qui permet, à partir des représentations vectorielles des entités, de les aligner. Apprendre la représentation d'une entité, revient à apprendre les caractéristiques qui la définissent, à savoir, les informations provenant à la fois de sa structure adjacente, et de ses attributs. Hamilton [74] propose dans son livre une vue synthétique des techniques d'apprentissage de représentation appliquées aux graphes. Nous renvoyons le lecteur vers ces deux articles [45, 75] pour de plus amples détails sur ces méthodes d'apprentissage de représentation appliquées au graphes.

Selon les représentations vectorielles (*embedding*) obtenues pour chacune des entités des graphes source et cible, deux entités sont dites similaires si la projection de leur représentation vectorielle au sein d'un même espace vectoriel est proche. En d'autres termes, la similarité entre deux entités est évaluée en fonction de la distance qui sépare leur représentation vectorielle. Généralement, évaluer la distance sémantique entre deux vecteurs de représentation, revient à mesurer le cosinus entre ces deux vecteurs [10].

Afin d'obtenir une représentation fiable du patrimoine informationnel d'une entité, différentes caractéristiques peuvent être extraites :

- *Les caractéristiques structurelles, à savoir les relations avec les entités adjacentes qu'une entité possède.* Les approches d'appariement et d'alignement, qui exploitent la structure des graphes,

partent du postulat que deux entités issues de deux graphes différents sont potentiellement similaires, si elles partagent le même voisinage, à savoir les mêmes relations et les mêmes entités adjacentes (entités de l'ordre 1). Par conséquent, la similarité sémantique entre deux entités repose sur les caractéristiques structurelles extraites.

– *Les caractéristiques propres, à savoir, les attributs et leur valeur, que possède une entité.* Exploiter les attributs des entités, ainsi que leur valeur, permet de formuler l'hypothèse selon laquelle deux entités équivalentes partagent des attributs similaires, ayant la même valeur. Ainsi, la similarité entre deux entités repose sur les caractéristiques propres extraites.

Les communautés spécialisées dans les thématiques d'appariement et d'alignement des entités (*entity matching, entity alignment, entity resolution*), et d'appariement et d'alignement des graphes (*graphs matching, graphs alignment*) sont des communautés très actives. Une revue détaillée de l'état de l'art de ces méthodes est présenté par Zhao *et al.* [196]. Différentes stratégies sont employées pour obtenir une représentation des entités suffisamment explicite pour favoriser leur alignement dans l'espace vectoriel latent, puis construire un mappage. Les approches d'appariement et d'alignements basées sur l'apprentissage de représentation des KG suivent généralement deux modèles :

– L'algorithme *TransE* appartient aux approches d'incorporation des graphes dans un espace vectoriel (*shallow embedding*). Il a vocation à apprendre une représentation vectorielle unique pour chaque entité.
– Les réseaux de neurones appliqués aux graphes (*Graph Neural Network* GNN).

TransE [22] est un modèle qui apprend la représentation vectorielle des entités et de leurs relations, en se basant sur des triplets en relation $\langle h, r, t\rangle$ où $h, t \in E$ et $r \in R$ (h, t et r étant définis par leur label). TransE permet de préserver les informations provenant de la structure voisine de chaque entité. De ce fait, les entités ont une représentation vectorielle qui dépend fortement des entités avec lesquelles elles partagent des relations. Par conséquent, deux entités qui partagent des relations avec des entités adjacentes similaires ont de grandes chances d'avoir une représentation vectorielle très proche l'une de l'autre, dans l'espace vectoriel latent appris par le modèle TransE. En d'autres termes la similarité sémantique entre deux entités dépend des relations qu'elles entretiennent avec d'autres entités. En plus de vectoriser les informations de la structure, Trisedya *et al.* [163], ainsi que Zhang *et al.* [195], proposent d'intégrer les informations provenant des attributs des entités. Trisedya *et al.* proposent d'exploiter les triplets en attribut $\langle h, r, v\rangle$, où $v \in V$ correspond à la valeur d'un attribut appartenant à une entité adjacente reliée par la relation $r \in R$. De leur côté, Zhang *et al.* proposent une approche multivue [110], qui permet d'apprendre la représentation vectorielle complète d'une entité en se basant sur plusieurs vues. De cette façon, une entité est décomposée en trois vues distinctes et complémentaires : (1) la vue à partir du nom de l'entité (label de l'entité) ; (2) la vue basée sur les relations de l'entité grâce aux triplets en relation $\langle h, r, t\rangle$; (3) la vue depuis les attributs de l'entité grâce aux triplets en attribut $\langle h, a, v\rangle$, où $a \in A$ est l'attribut de l'entité et $v \in V$ sa valeur. Les vues sont ensuite combinées pour obtenir une représentation plus précise qui favorise l'alignement de l'entité.

En comparaison, l'idée principale d'un GNN est de générer une représentation vectorielle d'une entité, qui dépend à la fois de la structure du graphe, mais également des informations que chaque

entité contient. Un GNN génère une représentation vectorielle pour chaque entité, en encodant les informations qui proviennent des entités adjacentes. Un mécanisme de transmission de messages vectoriels (ou *neural message passing* [66]) est utilisé pour propager les informations de chaque entité à travers la structure du graphe. Une mécanisme d'agrégation des informations vient ensuite compiler les informations d'une entité avec les informations qui proviennent de sa structure adjacente. Pour finir, le GNN joue ensuite le rôle de mécanisme d'actualisation de la représentation de chaque entité.

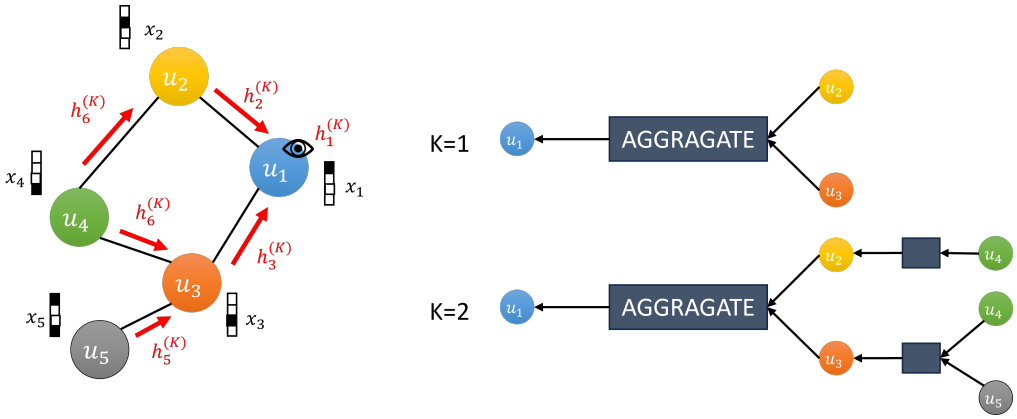


FIGURE 2.2 – Processus de *message passing* entre les entités

Suivant la Figure 2.2, le graphe de connaissance est composé des entités u_1, u_2, u_3, u_4, u_5, ayant respectivement des caractéristiques x_1, x_2, x_3, x_4, x_5 et un vecteur de représentation h_1, h_2, h_3, h_4, h_5. Portons notre attention sur l'entité u_1. Pour chaque itération K du GNN, le processus de *message passing* est enclenché, et un vecteur $h_u^{(k)}$, pour chaque entité du graphe, est actualisé en fonction des informations agrégées à partir du voisinage $\mathcal{N}(u)$ de l'entité u. Par conséquent, l'entité u_1 agrége les informations h_2 et h_3 provenant des entités u_2 et u_3.

Le processus de mise à jour de la représentation vectorielle d'une entité suit l'équation suivante [73] :

$$h_u^{(k+1)} = UPDATE^{(k)}(h_u^{(k)}, AGGREGATE^{(k)}(h_v^{(k)}, \forall v \in \mathcal{N}(u)))$$
$$h_u^{(k+1)} = UPDATE^{(k)}(h_u^{(k)}, m_{\mathcal{N}(u)}^{(k)})$$

où, les fonctions UPDATE et AGGREGATE sont réalisées par un réseau de neurones, $m_{\mathcal{N}(u)}^{(k)}$ est le "message", i.e., le résultat de l'agrégation des informations des entités adjacentes à l'entité u. Ainsi, pour chaque itération K, la fonction AGGREGATE prend en entrée l'ensemble des représentations vectorielles des entités adjacentes $\mathcal{N}(u)$ de l'entité u. La fonction UPDATE, quant a elle, combine le message $m_{\mathcal{N}(u)}^{k}$ avec la représentation de l'itération K précédente $h_u^{(K-1)}$ de l'entité u. De cette façon, chaque entité agrège les informations provenant de son voisinage local (d'ordre 1). Pour plus de précisions : Pour $K = 1$, la représentation vectorielle de chaque entité contient les informations des entités de l'ordre 1 ; Pour $K = 2$, la représentation vectorielle de chaque entité contient également les informations des entités de l'ordre 2 (voir Figure 2.2 pour $K = 2$).

Récemment, de nombreuses approches exploitent les atouts des modèles GNN pour améliorer l'apprentissage de représentation des entités et favoriser leur appariement et leur alignement dans l'espace vectoriel latent. Par exemple, dans le cas de l'alignement des entités de deux KG édités dans des langues distinctes (multilingue), Wang *et al.* [177] proposent par exemple d'exploiter un modèle neuronal à base de convolutions (*Graph Convolutional Network* GCN) [95], pour apprendre la représentation des entités dans un espace vectoriel unifié, qui permet ensuite de les aligner. L'avantage d'un GCN est qu'il encode à la fois la structure du graphe, mais également les informations de chaque entité. De ce fait, le modèle GCN prend ainsi en entrée les caractéristiques sous forme de vecteurs (ou *features*) de chaque entité, à savoir les informations de l'entité sous forme de label et de triplet en attribut $\langle h, a, v \rangle$, ainsi que la matrice d'adjacence, qui spécifie la structure du graphe, et les relations entre les entités, pour propager les *features* des entités à travers le graphe grâce au mécanisme de *message passing*. Liu *et al.* [112] proposent une approche où les graphes sources et cibles sont découpés en plusieurs sous-graphes qui regroupent des attributs de même type, à savoir, d'un côté les attributs ayant des valeurs numériques, et de l'autre côté les attributs ayant des valeurs textuelles, afin de simplifier l'alignement entre les entités. La représentation vectorielle de chacune des entités sera ensuite produite par un GCN pour chaque sous-graphe. La particularité de l'approche de Liu *et al.*, est qu'elle exploite un algorithme d'attention appliqué aux graphes [171], pour déterminer l'importance de chaque attribut, afin d'améliorer l'alignement des entités. De leur côté, Wu *et al.* [186] vectorisent à la fois le label des entités, et les relations typées (relations définies par un label) qu'elles entretiennent avec d'autres entités. Les auteurs partent du principe que les relations contiennent une source d'informations non négligeable, pour améliorer davantage l'opération d'alignement de deux entités.

Cependant, il est important de remarquer que la structure adjacente d'une entité n'est pas toujours isomorphique d'un graphe à l'autre. Il est souvent probable qu'une information à l'ordre 1 d'une entité dans un KG, soit présente à l'ordre 2, ou plus, dans un autre KG. C'est en ce sens que Sun *et al.* [156] proposent une approche permettant d'incorporer dans la représentation d'une entité, à la fois sa structure adjacente d'ordre 1 et d'ordre 2. Un système de porte est utilisé pour combiner les informations provenant de l'ordre 1 et de l'ordre 2. Un mécanisme d'attention est utilisé pour déterminer les entités adjacentes, qui présentent un plus grand intérêt pour l'alignement de deux entités.

Le Tableau 2.1 reprend l'ensemble des approches présentées précédemment. Il présente les caractéristiques extraites pour chacune des approches, ainsi que l'encodeur utilisé pour les encoder.

Approches	***Modélisation structure***	***Caractéristiques de l'entité***				***Encodeur caractéristiques***
		nom entités	*nom attributs*	*valeur attributs*	*nom relations*	
AttrE [163]	TransE	√	√	√		LSTM N-gram
MultiKE [195]	TransE	√	√	√		*word2vec* [118] *Skip Gram* [119]
GCN-Align [177]	GCN		√			
AttrGNN [112]	GCN	√	√	√		BERT [51]
HGCN [186]	GCN	√			√	

TABLE 2.1 – Synthèse des approches d'appariement et d'alignement appliquées aux graphes

2.1.1.2 Opération de mappage entre ontologies

Les ontologies sont des représentations conceptuelles formelles et explicites d'un domaine spécifique. Elles sont définies par un ensemble de concepts et leurs propriétés, ainsi que par des relations sémantiques entre ces concepts. Par définition, une ontologie spécifie la terminologie d'un domaine, puis organise et standardise l'arborescence de ses concepts les uns par rapport aux autres. Les ontologies sont couramment utilisées par les technologies du *Web Sémantique*, pour formaliser la connaissance et promouvoir l'échange et le partage de données. Depuis longtemps, les ontologies sont utilisées pour assurer l'interopérabilité sémantique entre deux sources de données. Cependant les ontologies décrivent toujours une partie très spécifique d'un domaine, ce qui les rend très souvent incomplètes et accroît la nécessité de les aligner et, ou, de les fusionner. Le domaine médical est un excellent exemple, les ontologies *International Classification of Diseases* (ICD), *Unified Medical Language System* (UMLS), ou encore *Systematized Nomenclature of Medicine-Clinical Terms* (SNOMEDCT), sont complémentaires et couvrent, pour la plupart, les même expertises. Aligner les concepts de ces différentes représentations permettrait d'obtenir une représentation d'autant plus détaillée et précise.

Depuis longtemps, des méthodes sont proposées pour tisser un mappage entre plusieurs ontologies. La plupart s'appuient sur des mesures de similarité structurelles, syntaxiques (couramment appelées *string-metrics*) et sémantiques.

Les mesures structurelles évaluent la similarité entre deux topologies d'ontologies. Étant donné que les ontologies sont structurées sous forme d'arbres contenant des super-concepts, des concepts et des sous-concepts, une approximation de la similarité des structures de deux ontologies peut être calculée. Les mesures basées sur la topologie quantifient les caractéristiques internes d'un concept, à savoir le nombre d'attributs et le nombre de méthodes qui lui sont propres, ou encore, les caractéristiques externes d'un concept, telles que les types de relations entretenues avec d'autres concepts, ainsi que les types de concepts qui lui sont adjacents [40].

Les mesures syntaxiques Les mesures syntaxiques évaluent les similitudes entre deux manières d'écrire un même mot. Très souvent, deux mots sémantiquement similaires peuvent être écrits de façons très différentes. Par exemple, suivre une convention d'écriture tel que le *CamelCase*, ou encore rajouter

un suffixe et un préfixe au nom d'un élément, peut drastiquement compliquer la création d'un mappage entre deux concepts. Des mesures de distance permettent d'évaluer si l'écriture de deux mots est proche ou non. Parmi ces mesures, la Distance de Levenshtein [190] (*Normalized Levenshtein Edit Distance*), est souvent utilisée. La Distance de Levenshtein, ou *LevNorm*, renvoie le nombre d'éditions nécessaires pour transformer une chaîne de caractères en une autre, en utilisant les opérations d'insertion, de suppression et de substitution pour chaque caractère. Dans leur article, Stoilos *et al.* [153] présentent les mesures de similarité syntaxiques majoritairement utilisées pour aligner deux concepts.

Les mesures sémantiques évaluent si deux concepts partagent la même signification. Généralement, pour évaluer si deux mots partagent le même sens, une base de connaissances est nécessaire pour interpréter le sens de chacun des mots et mesurer la distance sémantique qui les sépare. En principe, pour des activités d'appariement, la base lexicale la plus utilisée est *WordNet* [120]. Elle sert généralement d'ontologie de référence pour identifier si deux termes sont de même nature. Dans *WordNet*, les mots sont regroupés par synonymes dans des groupes nommés *synsets*. Selon le contexte, chacun de ces mots a un sens particulier. Par exemple, le mot *star* détient deux sens : l'un pour spécifier un objet céleste, l'autre pour désigner une célébrité. Chaque *synset* est relié à d'autres *synsets* par des relations sémantiques qui spécifient une certaine hiérarchie entre les groupes (par exemple, dans le cas d'hyponymes et d'hypernonymes). Ces relations sémantiques peuvent également traduire l'antonymie de deux mots dans le cas où leur sens serait opposé. Pour mesurer la distance sémantique entre deux mots, il est courant d'utiliser le *cosinus similarité* [10]. Le *cosinus similarité* mesure l'angle entre la représentation vectorielle d'un élément du métamodèle source et la représentation vectorielle d'un élément du métamodèle cible.

L'étude [82] démontre qu'utiliser une unique mesure de similarité pour aligner les concepts de deux ontologies est insuffisant, et préconise l'emploi de plusieurs indicateurs de similarité pour découvrir davantage de mappages. Parmi ces indicateurs, les mesures de similarité syntaxiques sont employées, ainsi que des mesures structurelles qui se basent sur la hiérarchie des concepts (chemin depuis l'élément de tête vers le concept étudié).

Les approches d'alignement de concepts peuvent être également étudiées sous l'angle d'un problème de classification par les algorithmes d'apprentissage machine. Dans leur étude, Nejhadi *et al.* combinent à la fois des mesures de similarité, ainsi que des algorithmes de classification tel que, le *K Nearest Neighbor* (KNN), le *Support Vector Machine* (SVM) et le *Decision Tree* DT.

Enfin, les récentes avancées dans le domaine de l'apprentissage profond ont permis d'améliorer considérablement le problème d'alignement des concepts entre plusieurs ontologies. Bento *et al.* [16] utilise un réseau de neurones convolutifs (*Convolutional Neural Networks* CNN), pour capturer les informations provenant de la structure de l'ontologie. Ils partent du principe que si deux concepts sont rattachés à des concepts enfants et parents similaires, alors ils peuvent être considérés comme équivalents. Selon Iyer *et al.* [85], les approches actuelles peinent à modéliser le contexte d'un concept. De ce fait, ils proposent une mesure de similarité structurelle qui se base sur le contexte spécifique que chaque concept détient. La modélisation du contexte d'un concept se fait selon deux points de vue : (1) Les relations d'ordre 1 entretenues avec les concepts adjacents ; (2) Les chemins depuis l'élément de tête vers le concept étudié. Un mécanisme d'attention permet ensuite de déduire le chemin ainsi

que le concept adjacent qui comportent le plus d'influence sur l'alignement du concept étudié.

Récemment, les approches d'apprentissage de représentation de graphes (développées dans la Section 2.1.1.1), ont prouvé leur grande capacité à apprendre la représentation vectorielle d'une entité en se basant sur son contexte, à savoir, les entités adjacentes avec lesquelles l'entité partage des relations. Hao *et al.* exploitent un GCN pour capturer la structure hiérarchique des concepts. Étant donné que, d'une ontologie à l'autre, la structure adjacente d'un concept est souvent non-isomorphique, les auteurs agrègent à la fois la structure locale (adjacente) et la structure globale de chaque concept. De leur côté, Xiang *et al.* [187] vectorisent conjointement la structure des deux ontologies, en utilisant le modèle TransE. Quant aux travaux de Wu *et al.* [185], les auteurs ont développé un encodeur attentionnel, basé sur le mécanisme d'attention appliqué aux graphes de [171], qui permet d'apprendre simultanément la représentation vectorielle d'un concept, à partir de sa structure et de sa terminologie, avant de procéder à son alignement dans l'espace vectoriel.

2.1.2 Opération de dérivation des règles de transformation

A présent, l'état de l'art détaillé dans cet axe sera centré sur les approches permettant d'établir des règles de transformation qui spécifient les relations structurelles et sémantiques entre deux métamodèles. Suivant la classification d'Alwan, trois approches seront analysées : (1) les approches d'apprentissage par l'exemple basées sur des instances de modèles (M1) (voir Section 2.1.2.1) ; (2) Les approches par appariement des métamodèles (voir Section 2.1.2.2) ; (3) Les approches hybrides exploitant une combinaison des deux dernières approches (voir Section 2.1.2.3). La Figure 2.3 synthétise l'ensemble des approches étudiées, ainsi que les méthodes et les algorithmes utilisés par chacune d'entre elles.

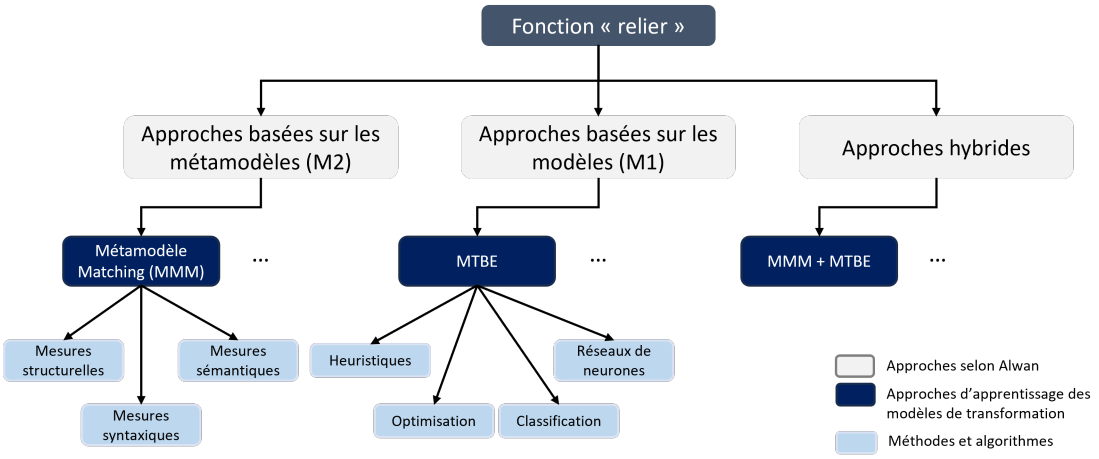


FIGURE 2.3 – Classification des approches permettant de réaliser la fonction "relier"

2.1.2.1 Approches d'apprentissage par l'exemple

Au premier abord, il semble parfois plus évident de fournir des instances de modèles provenant de deux domaines distincts, plutôt que de définir directement des règles de transformation au niveau de leur métamodèle pour passer d'un domaine à l'autre. C'est en ce sens que Varró [19] propose un nouveau champ d'étude, où l'apprentissage des modèles de transformation est guidé par l'analyse d'instances de modèles donnés à titre d'exemple. Depuis plusieurs années maintenant, de nombreuses approches ont vu le jour. Elles s'appuient principalement sur les travaux de Varró et exploitent des algorithmes heuristiques et d'optimisation, ainsi que des méthodes d'apprentissage machine et d'apprentissage profond pour automatiser l'apprentissage des modèles de transformation.

Initiée par Varró [19] la génération de modèles de transformation par l'exemple (*Model Transformation by Example* MTBE) est la première approche semi-automatique permettant de déduire les règles de transformation à partir d'un corpus de modèles (M1) donné en exemple. Ce corpus est constitué de paires de modèles issues du domaine source et cible et a vocation à guider l'apprentissage des règles de transformation entre le métamodèle du domaine source et le métamodèle du domaine cible. En complément, un pré-alignement, spécifiant les relations d'équivalences entre les éléments des modèles source et cible, est fourni par un expert pour assister le processus d'apprentissage. Les règles de transformation sont alors dérivées à partir de ce pré-alignement. Les règles générées entrent ensuite dans un processus itératif faisant intervenir un expert pour affiner le programme de transformation. L'objectif du processus itératif d'affinage est de se rapprocher le plus fidèlement possible du modèle cible souhaité, tout en minimisant les pertes sémantiques lors de la transformation d'un modèle source. L'approche initiale de Varró est par la suite implémentée par l'application des principes de la Programmation Logique Inductive (*Inductive Logic Programming* ILP), pour automatiser complètement le processus de génération de règles de transformation [14]. Dans la continuité des travaux de Varró, des approches similaires sont proposées pour traduire les règles obtenues par le processus MTBE en règles opérationnelles dans la syntaxe du langage de transformation ATL [183, 154]. Contrairement à Varró et al., [183, 154] jugent qu'il est plus évident pour un expert d'identifier des relations d'équivalence en se basant sur la syntaxe concrète des modèles plutôt que sur leur syntaxe abstraite.

Dolques et al. [53] proposent une approche par apprentissage automatique basée sur les méthodes du *machine learning*. Leur démarche suit les principes dictés par l'Analyse Relationnelle de Concepts (*Relational Concept Analysis* RCA) [81], qui permet la classification d'objets en fonction de leurs propriétés. L'approche RCA prend également en considération les objets décrits selon leurs relations avec d'autres objets. De ce fait, inférer les règles de transformation revient principalement à trouver les caractéristiques communes partagées par les éléments des modèles source et cible donnés à titre d'exemples. Pour initier leur démarche, un pré-alignement est préalablement réalisé manuellement pour établir les liens de transformation entre les éléments des modèles source et cible. Leur approche est résumée en trois étapes ; (1) la classification des éléments des modèles source et cible ; (2) la classification des relations d'équivalence préétablies ; (3) l'interprétation des résultats en règles de transformation. La démarche proposée par Saada *et al.* [145] est dans la continuité des travaux présentés par [53] et consiste à générer automatiquement des règles de transformation opérationnelles dans le langage de transformation JESS.

Par ailleurs, le problème d'apprentissage des modèles de transformation peut être abordé sous le point de vue d'un problème d'optimisation. Face au nombre de combinaisons possibles lorsqu'il s'agit d'apprendre les règles qui transforment les éléments du modèle source en éléments du modèle cible, il peut parfois s'avérer impossible d'aboutir à une solution optimale. Kessentini *et al.* proposent deux solutions pour explorer l'espace technique des solutions [91, 93]. La première est une approche basée sur les principes de l'Optimisation par Essaims Particulaires [91](*Particle Swarm Optimization* PSO). La seconde méthode combine la méthode PSO avec la méthode du Recuit Simulé (*Simulated Annealing* SA), pour affiner localement les règles produites par la première méthode. Parti du constat que l'espace de recherche des solutions est trop vaste lorsqu'il s'agit de résoudre des transformations complexes faisant intervenir des conditions de transformation, Baki et Sahraoui [13] proposent une approche multi-étapes. La première étape consiste à analyser les relations d'équivalence préalablement définies (le pré-alignement) qui associent les éléments des modèles source et cible donnés en exemple. Ces relations sont alors regroupées par catégories (*pool*), de sorte à réduire l'espace de recherche et à améliorer l'efficacité des algorithmes d'apprentissage. La seconde étape permet, à partir des catégories identifiées précédemment, d'appliquer des algorithmes génétiques afin d'associer chaque relation à une règle de transformation qui convertit le mieux les éléments du modèle source en éléments du modèle cible. Pour terminer, l'ensemble des règles sont fusionnées en un seul programme de transformation qui est ensuite affiné à l'aide de la méthode SA pour améliorer les règles avec des conditions plus complexes.

Contrairement à l'ensemble des approches présentées précédemment, les travaux de Faunes *et al.* [60, 61] ne reposent pas sur un pré-alignement défini par un expert. Leur proposition emploie des algorithmes évolutionnaires pour retrouver les règles de transformation entre un modèle source et un modèle cible donnés en exemple. L'objectif de cette approche est de générer un programme de transformation déclaratif (exécutable) et réutilisable. Suivant les principes établis par la programmation génétique, une population initiale de règles de transformation aléatoires évolue sur plusieurs générations par croisements et mutations (opérateurs génétiques), pour se rapprocher de la transformation souhaitée. L'évolution est guidée par une paire de modèles exemples (source et cible) qui donne une indication sur la façon dont la transformation doit se comporter.

Enfin, les récents travaux initiés par Burgueño *et al.* [32, 33] tirent profit des avancées dans le domaine du *machine learning*, et tout particulièrement des méthodes d'apprentissage supervisé pour la reconnaissance de patterns. Traduire une séquence de symboles dans différents domaines est une thématique bien connue par la communauté de l'apprentissage du langage naturel (*Natural Language Processing* NLP), notamment dans le cas d'un problème de traduction automatique du langage. Pour cela, Burgueño *et al.* ont construit une architecture encodeur/décodeur LSTM (*Long Short-Term Memory*) combinée avec un mécanisme d'attention qui est entraîné à prédire (dans notre cas transformer) de manière autonome un modèle cible à partir du modèle source. L'encodeur apprend la représentation du modèle source tandis que le décodeur apprend à reconstruire le modèle cible à partir de la représentation du modèle source. Les réseaux de neurones LSTM sont une extension des Réseaux de Neurones Récurrents (*Recurrent Neural Network* RNN), qui ont pour propriété une mémoire à long terme. De cette façon, la mémoire à long terme d'un réseau LSTM détient la capacité de se souvenir des règles de transformation précédemment apprises pour édifier des règles de transformation plus

complexes. L'entraînement de l'architecture proposée est guidé par un corpus de modèles sources et cibles.

Le Tableau 2.2 synthétise l'ensemble des informations concernant les approches présentées précédemment. Les spécificités de chaque approche, à savoir, les données d'entrées et de sorties (M_s pour les modèles sources, M_c pour les modèles cibles et P pour les pré-alignements), ainsi que les mesures de validation et les jeux de données sur lesquels ont été testés les algorithmes, y sont renseignées.

Approches	***Entrées***	***Sorties***	***Cas d'étude***	***Mesures validation***	***Algo/data disponible ?***
Heuristique [170]	$M_s + M_c + P$	Règles	*UML2Relational* [19]	?	✗
Heuristique [183]	$M_s + M_c + P$	Règles ATL	*UML2ER*	?	✗
ILP [14]	$M_s + M_c + P$	Règles	*UML2Relational* [19]	?	✗
RCA [53]	$M_s + M_c + P$	Patterns	*Asso2Responsibilities*	?	✗
RCA [145]	$M_s + M_c + P$	Règles JESS	*UML2Relational* [46]	Précision Recall	✗
PSO/PSO+SA [91, 93]	$M_s + M_c + P$	Modèle cible	*UML2Relational* [46]	Recall Fitness Temps	✗
GA [60, 61]	$M_s + M_c$	Règles JESS	*UML2Relationaln*[46]	Recall Fitness	✗
GA [13]	$M_s + M_c + P$	Rules	*UML2Relational* [46]	Précision Recall F-measure	✗
LSTM [33] encoder/decoder	$M_s + M_c$	Modèle cible	*Class2Relational* [6]	Fitness Temps	✓

TABLE 2.2 – Synthèse des approches MTBE

2.1.2.2 Approches par appariement des métamodèles

Contrairement aux approches MTBE qui requièrent à la fois un corpus de modèles sources et cibles (M1), ainsi que, pour certaines méthodes, des pré-alignements fournis par un expert, les approches par appariement des métamodèles reposent uniquement sur l'identification de règles de transformation entre les éléments d'un métamodèle source et d'un métamodèle cible. Procéder directement à l'alignement des concepts de deux métamodèles permettrait ensuite de déduire automatiquement les potentielles règles de transformation qui les relient. C'est ce que proposent les approches d'appariement. Ces algorithmes prennent généralement deux métamodèles en entrée et produisent des relations d'équivalence, ou mappage, entre leurs éléments, à savoir, leurs classes, leurs attributs et leurs relations. Le mappage généré est une opération communément appelée "alignement" et qui permet de mettre en relation deux ou plusieurs éléments provenant de deux métamodèles différents.

Le principal défi revient à palier aux problèmes d'hétérogénéité, lorsqu'il s'agit d'identifier deux

concepts sémantiquement similaires provenant de métamodèles différents. Cette hétérogénéité est induite par des langages de modélisation spécifiques à chaque domaine, et qui font intervenir des structures et des terminologies qui sont différentes d'un métamodèle à l'autre. De nombreuses approches ont été utilisées pour identifier des relations de correspondance entre les éléments de deux métamodèles. La plupart d'entre elles se basent sur le calcul d'un score évaluant la similarité entre deux éléments. Ces mesures évaluent les similarités structurelles, syntaxiques et sémantiques.

Dans leur étude, Falleri *et al.* [58] ont exploité l'algorithme de *Similarity Flooding* (SM) pour élaborer l'alignement entre deux métamodèles. Le principe sur lequel se base l'algorithme de SM est le suivant : "si deux concepts dans les métamodèles source et cible sont similaires, alors la possibilité que les concepts adjacents soient similaires augmente". Autrement dit, pour deux classes A et B similaires, les voisins de la classe A sont potentiellement similaires aux voisins de la classe B. De ce fait, l'algorithme de SM exploite pleinement la structure des métamodèles, en propageant des scores de similarités, par l'intermédiaire de relations entre les concepts des métamodèles source et cible. Pour initialiser l'algorithme, un simple opérateur de comparaisons calcule la similarité syntaxique entre le nom des éléments des métamodèles source et cible. Les éléments qui présentent un fort score de similarité sont alors couplés. L'alignement généré par l'algorithme de SM est ensuite utilisé pour produire un modèle de transformation indépendant de tout langage de transformation. Pour cela, les auteurs suivent les recommandations établies par Lopes *et al.* [113] en utilisant un métamodèle qui capitalise les relations d'alignement produites.

De leur côtés, Kessentini *et al.* [92] considèrent l'approche d'appariement des métamodèles comme un problème d'optimisation pour réduire l'espace de recherche des solutions et limiter les possibles combinaisons entre éléments des métamodèles source et cible. Dans un premier temps, ils utilisent un algorithme génétique pour générer aléatoirement une population initiale de potentielles relations de correspondance entre les éléments des métamodèles source et cible. Cette population est ensuite affinée par un algorithme de Recuit Simulé qui détermine les éléments du métamodèle source et du métamodèle cible qui partagent le plus de similarité. La similarité entre deux éléments est évaluée grâce à des mesures structurelles (qui caractérisent la topologie d'un élément en quantifiant les attributs, les méthodes et ses relations avec d'autres éléments), syntaxiques (qui déterminent la similarité des éléments par rapport aux noms des classes, des attributs, des méthodes et des associations) et sémantiques (qui déterminent la distance sémantique entre deux noms). La combinaison de ces trois mesures de similarité permet, *in fine*, de générer les relations de correspondance entre les éléments des métamodèles source et cible qui partagent un score de similarité élevé.

Concernant les travaux de Fang et Lano, l'approche [59] propose de mesurer la similarité entre les éléments des métamodèles source et cible, en se basant sur une représentation des classes dite "aplatie" [17]. De cette façon, chaque classe est représentée par les propriétés héritées (associations rattachées à une classe) et composées (attributs de la classe). La capture des propriétés héritées de la classe permet ainsi de préserver la structure du métamodèle. Pour identifier les relations de correspondance entre les deux métamodèles, Fang et Lano calculent ensuite un score de similarité pour chaque couple (classe source, classe cible), basé sur la structure des données (*Data Structure Similarty* DSS), à savoir sur la typologie des propriétés héritées et composées de la classe, ainsi que sur la terminologie des

classes (nom des classes). Lano et Fang étendent par la suite leur approche, en formalisant, une fois l'alignement entre les classes des métamodèles source et cible généré, des règles de transformation dans la syntaxe du langage ATL [102].

Dans leur étude, Wang *et al.* [176] proposent une méthodologie générique pour construire des modèles de transformation, afin d'assurer le partage de données entre systèmes hétérogènes. Le terme "générique", sous-entend que la méthode de construction du modèle de transformation est "universelle", c'est à dire non spécifique à un domaine particulier. A l'image de Kessentini *et al.* [92], ils combinent également des mesures sémantiques et syntaxiques pour évaluer la similarité entre les éléments des métamodèles source et cible, pour recréer automatiquement le mappage entre ces deux domaines. Pour y parvenir, ils exploitent la base de données lexicales *WordNet*. Addazi *et al.* [2] proposent également d'exploiter la base de données lexicales *WordNet* pour évaluer la similarité sémantique entre deux éléments, mais cette fois-ci, dans le cadre d'une évolution ou d'une modification de la topologie d'un unique métamodèle. De cette manière, l'ancien métamodèle est comparé au nouveau métamodèle. L'objectif final est de faire évoluer le modèle de transformation, pour qu'il considère de nouvelles règles de transformation.

Semblable aux approches [176, 2], les travaux de Schwichtenberg *et al.* [147] exploitent une ontologie de domaine en guise de base de connaissances pour traduire les messages échangés entre deux services (services producteur et consommateur). L'ontologie de domaine sert d'intermédiaire pour mesurer la correspondance sémantique entre les éléments des métamodèles des services producteur et consommateur appartenant à une Architecture Orientée Services (*Service-Oriented Architectures* SOA). Pour chaque paire de classes et d'attributs un score de dissimilarité est calculé. Ils déterminent de cette façon le mappage entre les attributs et les classes des différents métamodèles. Le mappage obtenu est ensuite traduit dans la syntaxe du langage de transformation QVT.

2.1.2.3 Approches hybrides

Les approches hybrides couplent à la fois les avantages de l'approche MTBE et de l'approche par d'appariement des métamodèles. Les approches MTBE sont pratiques pour analyser le comportement d'un élément du métamodèle source une fois transformé dans la syntaxe du métamodèle cible. L'étude des instances des modèles source et cible permet d'identifier des règles de transformation entre les modèles, de sorte à les généraliser ensuite au niveau métamodèle. A l'inverse, les approches par appariement sont intéressantes dès lors que les instances de modèles sont rares, ou, lorsque les données qui proviennent des instances de modèle sont très proches d'un point de vue structurel, syntaxique et sémantique.

C'est en ce sens que Dolques *et al.* [52] proposent une approche permettant d'automatiser la création des pré-alignements nécessaires aux approches MTBE, en appliquant des mesures structurelles et syntaxiques aux instances de modèles données à titre d'exemples pour guider l'apprentissage. Le processus de d'appariement utilisé s'appuie sur l'approche *Anchor-Prompt* [129], qui, dans un premier temps, permet de découvrir un à un les paires d'éléments source et cible (e_s, e_c) qui partagent un haut degré de similarité (paires appelés *anchors*), puis dans un second temps, le score de similarité de l'*anchor* est propagé à travers le modèle par l'intermédiaire des relations. La propagation du score de

similarité d'un *anchor* permet de découvrir de nouveaux mappages plus complexes. De ce fait, Dolques *et al.* montrent qu'il est pertinent de coupler les approches d'appariement pour générer un mappage, avec une approche MTBE pour améliorer la qualitié et l'automatisation de l'apprentissage des règles de transformation.

L'approche de Lano *et al.* [101] combine également à la fois l'approche par appariement des métamodèles et l'approche MTBE. Dans leurs travaux, Lano *et al.* optent pour une approche qui s'appuie sur les algortihmes de NLP, pour traiter l'énoncé des exigences du modèle de transformation, puis automatiser la création des relations de correspondance entre les métamodèles source et cible. L'énoncé des exigences [162] définit les spécifications que doit respecter le modèle de transformation. Les auteurs couplent ensuite les principes de la logique inductive avec l'approche MTBE pour corriger et détailler davantage le mappage réalisé précédemment. Leurs résultats prouvent que ces deux approches sont complémentaires et permettent d'inférer automatiquement des modèles de transformation plus fiables.

2.2 Synthèse de l'état de l'art

2.2.1 Un pré-alignement entre les modèles source et cible

La plupart des travaux appliquant l'approche MTBE reposent essentiellement sur des pré-alignements, i.e. des relations de correspondances manuellement préétablies entre les éléments des modèles source et cible. Ce pré-alignement n'est pas toujours évident à fournir, il nécessite souvent un heuristique métier. D'autre part, comme souligné par Burgueño *et al.* [32] la majeure partie des propositions vise à simplifier la définition des modèles de transformation, sans pour autant totalement l'automatiser. Force est de constater que l'impact de l'humain est omniprésent dans le processus de description des modèles de transformation, le manque d'automatisation est une lacune significative qui incombe au développement d'un mécanisme de connexion à la demande, *plug and play*.

Cependant, dans certains cas, les relations de correspondance peuvent être déduites automatiquement, sans intervention de l'humain, grâce à l'examen de la structure et de la terminologie des modèles source et cible. Ainsi, le nom d'un élément du modèle source est souvent similaire, ou quasiment égal, au nom de l'élément du modèle cible qu'il a généré après transformation. Il en va de même pour la structure des modèles. Le voisinage d'un élément e du modèle source est souvent transformé en un voisinage de l'élément transformé e' du modèle cible. De cette façon, l'élément source et l'élément cible qui en résulte après transformation, partagent, potentiellement, les mêmes voisins. De plus, au niveau instance des modèles (niveau data), la valeur d'un attribut du modèle source est souvent très proche de la valeur du même attribut dans le modèle cible. La valeur des attributs ne change généralement pas.

Les approches hybrides semblent être une solution intéressante pour compenser le manque de pré-alignement. Les travaux présentés par Dolques *et al.* [52] combinent à la fois les techniques d'appariement, pour retrouver automatiquement les relations de correspondances et établir un pré-alignement entre les éléments des modèles source et cible, et l'approche MTBE permettant de dériver, à partir du pré-alignement, les règles de transformation qui convertit un élément du métamodèle source en un

élément du métamodèle cible. De la même façon, Lano *et al.* [101] infèrent, dans un premier temps, un pré-alignement entre les éléments des métamodèles source et cible grâce à l'analyse linguistique de l'énoncé des exigences, puis, dans un second temps, utilisent l'approche MTBE pour affiner le mappage ainsi que les règles de transformation.

Pour améliorer la qualité de l'apprentissage des règles de transformation, il semble systématiquement nécessaire de fournir un pré-alignement, à savoir, un mappage entre les éléments des métamodèles source et cible. Les opérations de mappage entre les graphes (voir Section 2.1.1.1), les ontologies (voir Section 2.1.1.2) et les métamodèles (voir Section 2.1.2.3) semblent être des solutions intéressantes pour obtenir un mappage suffisamment cohérent en vue de dériver les règles de transformation entre deux métamodèles.

2.2.2 Des mesures de similarités

Concernant les approches qui exploitent des mesures de similarités pour identifier si deux éléments sont identiques, de nombreuses limites ont été identifiées. Tout d'abord, lors du processus de d'appariement, tous les scores de similarité sont calculés un à un pour chaque paire d'éléments source et cible. De ce fait, lorsqu'il s'agit d'établir le mappage entre deux métamodèles qui comptent un nombre élevé de concepts et d'éléments, l'espace des solutions augmente, ce qui signifie que le temps de computation augmente également. Par ailleurs les mesures de similarités sont inefficaces, dans le cas où les métamodèles source et cible auraient une structure et une terminologie très éloignées l'une de l'autre. D'autre part, lors de mesures sémantiques, l'utilisation d'une base de données lexicales trop générale, telle que *WordNet*, ne permet d'identifier des similarités sémantiques entre deux métamodèles très spécialisés dans un domaine particulier. Enfin, la combinaison de plusieurs mesures de similarité est une activité complexe. Comment prioriser une mesure plutôt qu'une autre ? En effet, la combinaison de mesures de similarité implique toujours de fixer, au préalable, des coefficients qui indiquent l'importance de telle ou telle mesure par rapport aux autres. Or, dans le cas où les règles de transformation ne sont pas connues à l'avance, comment identifier les bons coefficients ?

2.2.3 Une application des réseaux de neurones

Malgré les résultats très prometteurs présentés par Burgueño *et al.* [32, 33], une approche basée sur l'apprentissage des modèles de transformation par un réseau de neurones semble inapplicable dans le cadre de l'Industrie 4.0. D'un point de vue technique, un réseau de neurones artificiels requiert un grand nombre de données pour parfaire son entraînement (1000 modèles sont nécessaires pour obtenir une prédiction précise à 1.0). D'autant plus que les mappages complexes soumis à des conditions et de types n-m nécessitent davantage de données d'entraînement. La diversité du jeu d'entraînement est un autre facteur limitant, qui porte préjudice à l'approche énoncée par Burgueño *et al.*. Le réseau de neurones artificiels est capable de prédire uniquement les scénarios qu'il a, auparavant, appris. Enfin la durée de l'entraînement dépend grandement du nombre de modèles (pour 4000 modèles, la durée de l'entraînement est de 483 secondes), mais aussi du nombre d'éléments par modèles (pour 30 éléments dans un modèle, la durée de l'entraînement est de 4060 secondes).

2.2.4 L'inexistence d'un cadre expérimental

Comme le prouve le Tableau 2.2, les approches MTBE sont souvent validées par des indicateurs de performance différents, et ne sont pas toujours appliquées au même jeux de données de validation. Mesurer l'amélioration de la qualité concernant les approches MTBE n'est pas une activité aisée. Il est difficile de cerner les avantages, les lacunes et les avancées réalisées par les approches les plus récentes. Un cadre commun de compréhension est nécessaire pour simplifier, et surtout permettre, la comparaison des anciennes et des futures approches, de sorte à mettre en évidence leurs avantages et leur faiblesses vis-à-vis des exigences de l'Industrie 4.0.

A l'image de l'*Ontology Alignment Evaluation Initiative*[1](OAEI), qui permet la réalisation d'études comparatives pour mesurer la qualité des approches proposées permettant de résoudre des problèmes d'alignement sur des jeux de données standardisés et catégorisés, un protocole d'expérimentation similaire serait propice à la comparaison des approches d'apprentissage des transformations de modèles.

2.3 Orientation du sujet de recherche

D'après la synthèse de l'état de l'art précédente, il semblerait que l'opération de mappage soit une étape inévitable pour permettre la dérivation automatique des règles de transformation. Ce mappage peut être édifié à plusieurs niveaux : métamodèle (M2), modèle (M1), ou, les deux niveaux à la fois. Les approches par appariement des métamodèles permettent, dans un premier temps, d'établir un mappage entre les éléments de plusieurs métamodèles puis, dans un second temps, de dériver ces relations en règles de transformation exécutables. Les approches MTBE raisonnent sur un mappage entre deux instances de modèles préalablement identifié par des experts, pour éditer des règles de transformation. L'opération de dérivation des règles de transformation dépend fortement de la qualité du mappage identifié, tandis que la qualité du mappage dépend majoritairement de la capacité à capturer la représentation fidèle des concepts en fonction du contexte dans lequel ils interviennent.

Étant donné que l'objet d'étude porte sur les modèles de transformation permettant de définir des règles de transformation entre plusieurs métamodèles, et que la proposition fondamentale de l'étude concerne l'apprentissage des modèles de transformation par des mécanismes d'apprentissage machine ou d'apprentissage profond, nous proposons d'apprendre, conjointement, le mappage entre deux instances de modèles, ainsi que les règles de transformation qui lient deux métamodèles, pour automatiser la construction d'un modèle de transformation. Cabot *et al.* [34], dressent les bénéfices que pourraient engendrer, pour une architecture MDA, les approches qui exploitent les avancées du domaine de l'Intelligence Artificielle, et plus particulièrement les méthodes d'apprentissage machine. Cependant, les algorithmes d'apprentissage machine requièrent une grande quantité de données d'entraînement, pour améliorer la qualité de leur prédiction. Lorsqu'il s'agit de traduire, ou dans le cas présent, de transformer des modèles, il est nécessaire de fournir un large corpus d'exemples de modèles source et cible [126, 33]. Partant de ce constat, les opérations de mappage et de dérivation des règles de transformation suivront les principes de l'Apprentissage par Renforcement, et plus particulièrement

1. `http://oaei.ontologymatching.org/`

de l'apprentissage Q (*Q-learning*) [157]. L'avantage des méthodes d'apprentissage par renforcement est qu'elles assurent l'entraînement d'un agent pour une tâche spécifique avec une faible quantité de données. L'apprentissage est réalisé à travers les interactions que mène un agent intelligent avec son environnement. L'objectif de l'agent est d'apprendre la politique optimale π^*, qui maximise le total de récompenses cumulatives, suivant une séquence d'action réalisable. Dans le cas d'une transformation, la politique optimale consiste à formuler la suite de règles de transformation, à partir du mappage établi simultanément, qui permettent de transformer un modèle source en un modèle cible. L'objectif consiste à inférer, grâce aux techniques d'apprentissage Q, les liens qui relient les concepts d'un métamodèle source aux concepts d'un métamodèle cible, pour générer automatiquement un modèle de transformation. *In fine*, afin de répondre aux prérequis d'une interopérabilité dynamique, *plug an play*, la fonction de transformation apprise doit être réutilisable et applicable de manière autonome et automatique, quels que soient les modèles à transformer. Pour garantir la généricité de l'approche, le modèle de transformation inféré est indépendant de tout langage de transformation.

L'avantage d'exploiter les techniques d'apprentissage par renforcement réside dans leur capacité à optimiser le processus de décision dans sa globalité, sans pour autant optimiser les décisions individuellement. Considérant le problème de transformation de modèles, cela signifie que l'agent cherche à identifier la meilleure suite de règles qui permettra de transformer un modèle source en un modèle cible. En d'autres termes, l'agent se base sur la cohérence entre les décisions passées, à savoir les règles précédemment établies, et les décisions futures, c'est à dire les règles qui vont être établies. A l'image de l'approche [193], la notion de cohérence prend en considération l'interdépendance entre les règles précédemment établies, et les nouvelles règles qui seront définies par l'agent.

De récentes approches ont parfaitement exploitées les techniques d'Apprentissage par Renforcement pour des problèmes d'ingénierie dirigée par les modèles, notamment pour la réparation des modèles [15], et pour actualiser les modèles de transformation en prenant en compte leur évolution [57]. Nous suivons leur élan et proposons une approche d'apprentissage par renforcement des modèles de transformation.

2.4 Collection expérimentale

Une collection expérimentale recense l'intégralité des approches dédiées à un objet d'étude et répondant à une proposition fondamentale. Elle contient à la fois les algorithmes et les technologies développées par des travaux antérieurs, ainsi que les jeux de données qui ont servis de test et de validation. Étant donné que l'étude proposée dans ce manuscrit porte spécifiquement sur l'opération de dérivation des modèles de transformation, et que notre approche s'appuie principalement sur des instances de modèles (M1) pour inférer des règles de transformation, la collection expérimentale sera composée des approches MTBE présentée dans la Section 2.1.2.1. Cependant, comme la proposition fondamentale s'intéresse particulièrement aux approches d'apprentissage des modèles de transformation exploitant les techniques de l'intelligence artificielle, les approches utilisant des heuristiques métiers [170, 183] et ne générant par de règles de transformation [53], ne seront pas inclues dans la collection expérimentale.

Le Tableau 2.3 recense l'ensemble des approches MTBE retenues. Notons que ces approches seront ensuite utilisées dans le cadre d'une étude comparative de type *benchmark*, pour mesurer l'amélioration

de la qualité, que l'approche proposée dans ce manuscrit permettra d'atteindre.

Approches	***Algorithmes***	***Jeux de données***
Balogh *et* Varró [14]	ILP	*UML2Relational* [19]
Saada *et al.* [145]	RCA	*UML2Relational* [46]
Kessentini *et al.* [91, 93]	PSO/PSO+SA	*UML2Relational* [46]
Faunes *et al.* [60, 61]	GA	*UML2Relational*n[46]
Baki *et* Sahraoui [13]	GA	*UML2Relational* [46]
Burgueño *et al.* [33]	LSTM encoder/decoder	*Class2Relational* [6]

TABLE 2.3 – Collection expérimentale

2.5 Synthèse

Dans ce second chapitre, l'état de l'art de la fonction "relier", dont l'objectif est de tisser des relations structurelles et sémantiques entre plusieurs métamodèles, a été présenté. L'automatisation des modèles de transformation peut être divisée en deux étapes : (1) Une opération de mappage qui vise à relier les concepts qui partagent un sens commun ; (2) La dérivation des règles de transformation à partir du mappage.

Une synthèse de l'état de l'art concernant les approches de dérivation des règles de transformation, à savoir les approches d'appariement des métamodèles et les approches MTBE, a permis de mettre en évidence les avantages et les lacunes des propositions actuelles. A partir de cette analyse, nous avons fait le choix de positionner notre orientation de recherche parmi les approches MTBE. Plus précisément, le mappage des concepts des métamodèles, ainsi que la dérivation des règles qui lient les concepts, seront réalisés simultanément et seront basés sur l'analyse des modèles source et cible donnés à titre d'exemple. Le processus d'apprentissage suivra les principes de l'apprentissage Q.

Étant donné que l'orientation du sujet de recherche s'imbrique dans la catégorie des approches d'apprentissage par l'exemple, alors la collection expérimentale contiendra les algorithmes et les jeux de données des approches MTBE. La Figure 2.4 actualisée, instancie une collection expérimentale, dans l'objectif de mener une étude comparative de type *benchmark* dans la suite de ce manuscrit.

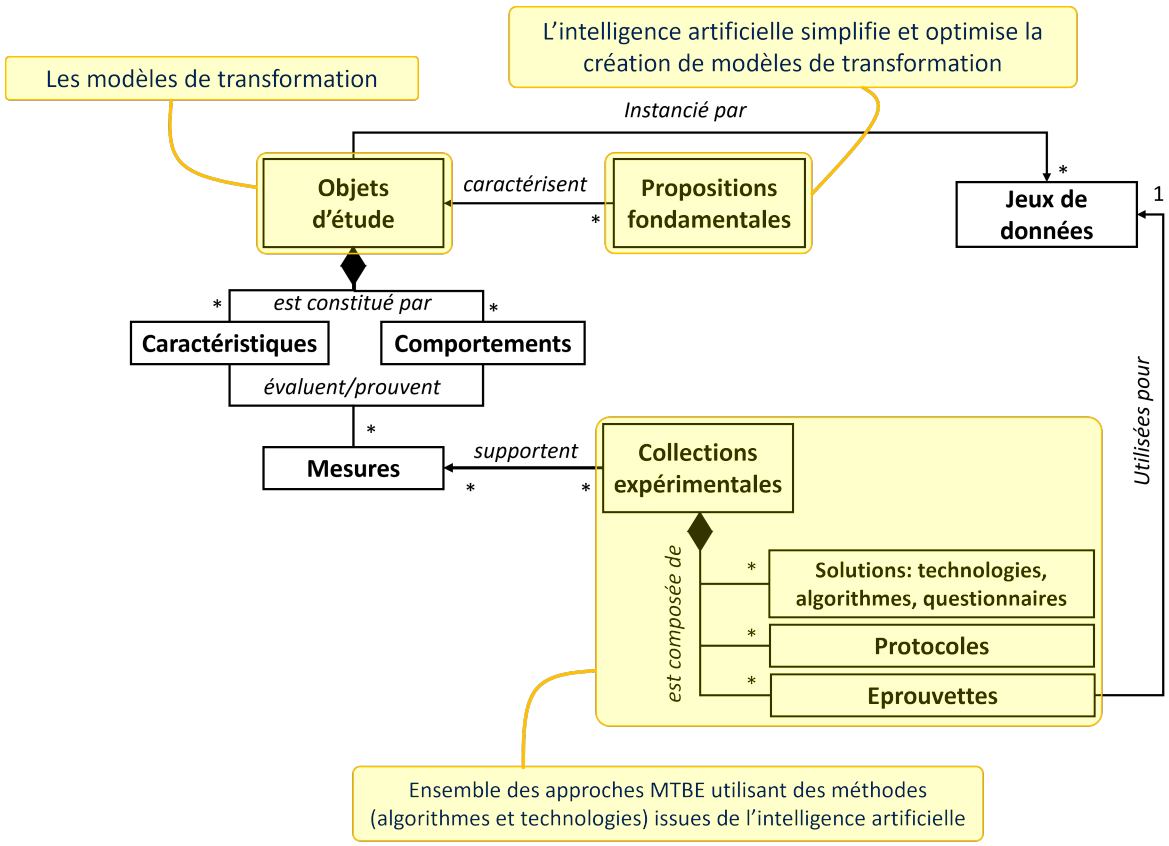


FIGURE 2.4 – Synthèse du Chapitre 2

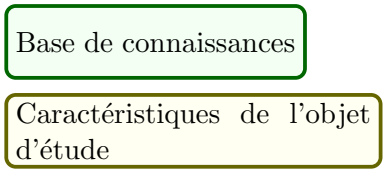
Chapitre 3

Proposition

Contenu

Grâce à la synthèse de l'état de l'art proposée au Chapitre 2, l'orientation du sujet de recherche a pu être définie. L'objectif est d'apporter des solutions concrètes aux lacunes identifiées des approches MTBE de la collection expérimentale. L'approche proposée consiste à apprendre, simultanément, le mappage entre les concepts des métamodèles source et cible et à dériver les règles de transformation qui lient ces concepts. Dans un premier temps, selon la méthodologie de recherche d'Hevner [80], la base de connaissances, qui a permis d'édifier la proposition de ce manuscrit sera présentée afin d'apporter des éléments théoriques sur les méthodes et les algorithmes employés. L'objectif de cette section est de permettre aux lecteurs de mieux appréhender la suite de ce chapitre (voir Section 3.1). Dans un second temps, l'approche d'apprentissage automatique, par renforcement, des règles de transformation sera ensuite présentée (voir Serction 3.3).

Cependant, chacune des approches de la collection expérimentale est appliquée à des ensembles de données spécifiques, afin de valider leur capacité à dériver automatiquement des règles de transformation correctes. Le Tableau 2.3 précédent, regroupe l'ensemble des jeux de données de validation utilisés. Il est important de se demander comment choisir le bon jeu de données pour valider l'approche proposée dans ce manuscrit ? Il est ensuite légitime de se demander comment comparer et mesurer l'amélioration de la qualité des approches d'apprentissage de modèles de transformation de la collection expérimentale ? Compte tenu de l'hétérogénéité des métamodèles, il n'est pas toujours facile de comparer les approches et de mesurer les améliorations de la qualité, car un modèle de transformation peut englober une variété de patterns de transformation. C'est en ce sens que les caractéristiques qui définissent un modèle de transformation seront décrites. Une grille d'évaluation sera proposée à la communauté scientifique pour faciliter la caractérisation des patterns de transformation qui interviennent lors d'une transformation de modèles (voir Section 3.2).



3.1 Base de connaissances

La base de connaissances fournit le socle de référence de toutes démarches de recherche. Elle pose les grandes lignes des méthodes utilisées pour parvenir à apporter une réponse concrète aux problématiques de la proposition fondamentale définie précédemment. Ces méthodes permettront d'édifier une solution concrète, afin de répondre aux grands défis de l'interopérabilité des systèmes, à savoir, le développement d'un mécanisme de connexion générique, automatique et dynamique entre les systèmes.

3.1.1 Introduction à l'apprentissage par renforcement

Étant donné que l'algorithme d'apprentissage des règles de transformation repose sur les principes de l'apprentissage par renforcement, cette première sous-section sera dédiée à l'apprentissage par renforcement.

3.1.1.1 Principes

Considérons un agent numérique qui évolue dans un environnement prédéfini et qui apprend des interactions qu'il réalise dans ce dernier en suivant le principe de tâtonnement. Pour chaque état $s_t \in \mathcal{S} = 1, \ldots, K$, l'agent sélectionne une action a_t appartenant à un ensemble d'actions prédéfini $a_t \in \mathcal{A} = 1, \ldots, K$ pour explorer son environnement. L'agent est alors projeté dans un nouvel état s_{t+1} qui dépend de l'action qu'il vient de commettre. En retour, ses actions sont gratifiées d'une récompense r_{t+1} plus ou moins élevée en fonction de l'état atteint et du changement opéré dans l'environnement.

La Figure 3.1 ci-dessous, présente un exemple concret de problème résolvable par les principes de l'apprentissage par renforcement. L'environnement est composé de quatre états, soit quatre cases, dans lesquelles l'agent pourra se rendre. Les actions possibles (haut, bas, gauche, droite) permettent à l'agent de passer d'une case à l'autre. Selon la case atteinte, une récompense de +1 est acquise si l'agent rejoint sa maison, tandis qu'une récompense de -1 est obtenue dans le cas où l'agent est obligé d'escalader un mur pour se rendre chez lui.

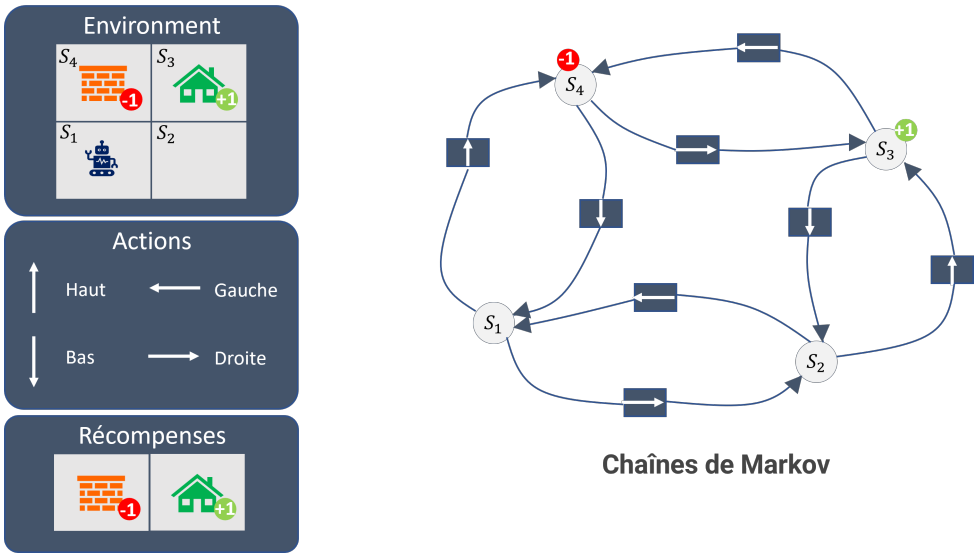


FIGURE 3.1 – Principes de l'apprentissage par renforcement

L'objectif de l'agent est d'apprendre la politique optimale $\pi^*(s_t)$, qui, selon un état (s_t), se résume à appliquer l'action a_t qui maximise l'espérance des récompenses dans le futur. L'équation de Bellman permet d'estimer la valeur d'état optimale tel que :

$$\pi^*(s_t) = r_{t+1} + \gamma \underset{a_{t+1}}{max} Q_i^*(s_{t+1}, a_{t+1}) \forall s_t \in \mathcal{S}$$

où la valeur Q optimale du couple état-action notée $Q^*(s_t, q_t)$ correspond au total cumulatif des récompenses futures avec rabais γ que l'agent peut espérer obtenir en sélectionnant l'action a_t pour un état s_t, tel que :

$$Q^*(s_t, q_t) = r_{t+1} + \gamma \underset{a_{t+1}}{max} Q_i^*(s_{t+1}, a_{t+1})$$

où r_{t+1} est la récompense que l'agent obtient lorsqu'il passe de l'état initial (s_t) à l'état suivant (s_{t+1}), après avoir réalisé l'action a_t, $\underset{a_{t+1}}{max} Q_i^*(s_{t+1}, a_{t+1})$ est la valeur Q optimale pour l'action a_{t+1} qui maximise l'espérance des récompenses à l'état s_{t+1}, et γ est le taux de rabais qui donne l'importance portée à l'espérance des récompenses futures.

En d'autres termes, l'agent apprend la stratégie optimale qui lui permettra de cumuler le maximum de récompenses en parcourant la chaîne de Markov, i.e. en exécutant une séquence d'actions possibles. Dans l'exemple précèdent, l'agent devrait déduire que la politique optimale pour résoudre son environnement, selon l'état initial de l'agent S_1, se résume par l'application de la suite d'actions "Droite" et "Haut". La chaîne de *Markov* [168] présentées dans la Figure 3.1 modélisent et formalisent le processus décisionnel appris par l'agent. De cette façon, selon un état s_t, pour chaque action possible, un coefficient de probabilité évaluera si il est bon d'utiliser une action plutôt qu'une autre. Par exemple, pour l'état S_1, deux actions sont possibles, les actions "haut" et "droite". Le coefficient de probabilité de l'action "droite" est certainement plus important que le coefficient de probabilité de l'action "haut" qui conduira à la récompense -1.

3.1.1.2 Politique d'exploration et d'exploitation

Lorsque l'agent démarre son apprentissage, l'ensemble des valeurs des couples état-action est initialisé à zéro. L'agent ne dispose d'aucune connaissance préalable pour résoudre son environnement, si ce n'est les états et les actions possibles.

Pour actualiser le poids de chaque couple état-action, une politique d'exploration de l'environnement est nécessaire pour permettre à l'agent de visiter l'ensemble des états S possibles. Il est important que l'agent puisse essayer chaque action possible plusieurs fois, pour accroître ses connaissances de l'environnement au fur et à mesure de ses expériences. Une approche consiste à suivre la politique *ϵ-greedy*, qui signifie que l'agent peut à la fois explorer son environnement en choisissant des actions aléatoires avec la probabilité ϵ, et exploiter les connaissances acquises précédemment pendant son apprentissage avec la probabilité $1 - \epsilon$ (soit en choisissant la valeur Q la plus élevée pour le couple état-action). Au fur et à mesure des itérations de l'algorithme, la valeur de ϵ diminue et pousse l'agent à agir de façon optimale en exploitant ses connaissances.

Au cours de sa progression dans l'environnement, l'agent optimise de façon itérative les valeurs Q pour chaque couple état-action en fonction des transitions et des récompenses obtenues, et construit petit à petit sa représentation de l'environnement sous la forme de chaînes de *Markov*. Il applique la formule d'apprentissage par Différence Temporelle (*Temporal Difference* TD) de *Bellman* pour actualiser le poids de chaque expérience,

$$Q_{i+1}(s_t, a_t) = Q_i(s_t, q_t) + \alpha[r_{t+1} + \gamma \underset{a_{t+1}}{max} Q_i^*(s_{t+1}, a_{t+1}) - Q_i(s_t, q_t)]$$

où i correspond au numéro de l'itération, $Q_{i+1}(s_t, a_t)$ est la valeur Q optimisée pour le couple état-action, α est le taux d'apprentissage, $r_{t+1} + \gamma \underset{a_{t+1}}{max} Q_i^*(s_{t+1}, a_{t+1})$ correspond à la valeur cible à atteindre pour l'itération i, et $Q_i(s_t, q_t)$ est la valeur Q obtenue pour avoir exécuté l'action a_t à l'état

s_t à l'itération i. Ensemble, $r_{t+1} + \gamma \underset{a_{t+1}}{max} Q_i^*(s_{t+1}, a_{t+1}) - Q_i(s_t, q_t)$ forment l'erreur TD à optimiser durant l'apprentissage. Concernant le facteur de rabais $\gamma \in [0; 1]$, plus γ est proche de 1, plus l'agent accorde de l'importance aux récompenses futures par rapport aux récompenses immédiates.

3.1.2 Jeux de données expérimentaux : Transformations *Class2Relational* et *Families2Persons*

Pour illustrer l'approche proposée, les transformations bien connues *Class2Relational* [6] (voir Figure 3.2) et *Families2Persons* [7] (voir Figure 3.3) mises à disposition par le zoo [1] ATL de la fondation Eclipse seront utilisées à titre d'exemple tout au long de ce manuscrit. Plusieurs raisons ont motivé le choix de ces deux transformations : (1) Les métamodèles, ainsi que les règles de transformation qui les relient, sont formellement explicites et disponibles sur le zoo ATL ; (2) Ces deux transformations de modèles font intervenir des patterns de transformation intéressants à étudier qui prouvent toute la complexité, ainsi que la variété, des patterns de transformation.

Les Tableaux 3.1 et 3.2 décrivent (en pseudo-code), respectivement, les règles de transformation qui interviennent lors des transformation *Class2Relational* et *Families2Persons*. Généralement, ces règles de transformation, dites descriptives, sont éditées à la main par un expert, et requièrent une connaissance détaillée des métamodèles source et cible pour tisser des relations structurelles et sémantiques entre leurs concepts. Ne répondant pas aux critères d'une interopérabilité *plug and play*, nous tenterons de retrouver ces règles de transformation à l'aide des méthodes et des algorithmes d'apprentissage machine pour simplifier et automatiser la génération des transformations de modèles.

Les éléments qui interviennent lors des transformations *Class2Relational* et *Families2Persons* seront présentés sous l'angle du méta-métamodèle *Ecore*, où, les classes correspondent à des éléments de type *EClass*, les associations entre les classes sont équivalentes à des éléments de type *EReference* et les attributs d'une classe sont représentés par des éléments de type *EAttribute*. Par exemple, le métamodèle d'un diagramme de classes est constitué d'éléments *EClass* de type [*Class*], eux même composés d'éléments *EClass* de type [*Attribute*] par l'intermédiaire d'une association de type *attr* correspondant à un élément *EReference*. L'élément de type [*Attribute*] est associé, soit à un élément de type [*Class*], soit à un élément de type [*DataType*] par l'intermédiaire d'un élément *EReference* de type *type*. Les éléments de type [*Class*] sont composés d'éléments *EAttribute* de type *name* et *isAbstract* (*True* ou *False* si la classe est abstraite ou non), tendis que les éléments de type [*Attribute*] sont composés d'éléments *EAttribute* de type *name* et *multiValued* (*True* ou *False* si l'attribut est un tableau de valeurs). Les éléments de type [*DataType*] sont composés d'un élément *EAttribute* de type *name*.

Concernant la transformation *Class2Relational* :

- *Modèle de classes.* Le modèle de classes est constitué d'une classe de type [*Class*] *Book* qui est elle même composée de plusieurs classes de type [*Attribute*] *Title*, *Editor* et *Authors*. Étant donné qu'un livre peut être rédigé par plusieurs auteurs, la valeur de l'attribut de type *multiValued* est égale à *True*. Les attributs *Title*, *Editor* et *Authors* sont associé à la classe de type [*DataType*] *String* car ce sont des valeurs textuelles.

1. `https://eclipse.dev/atl/atlTransformations/`

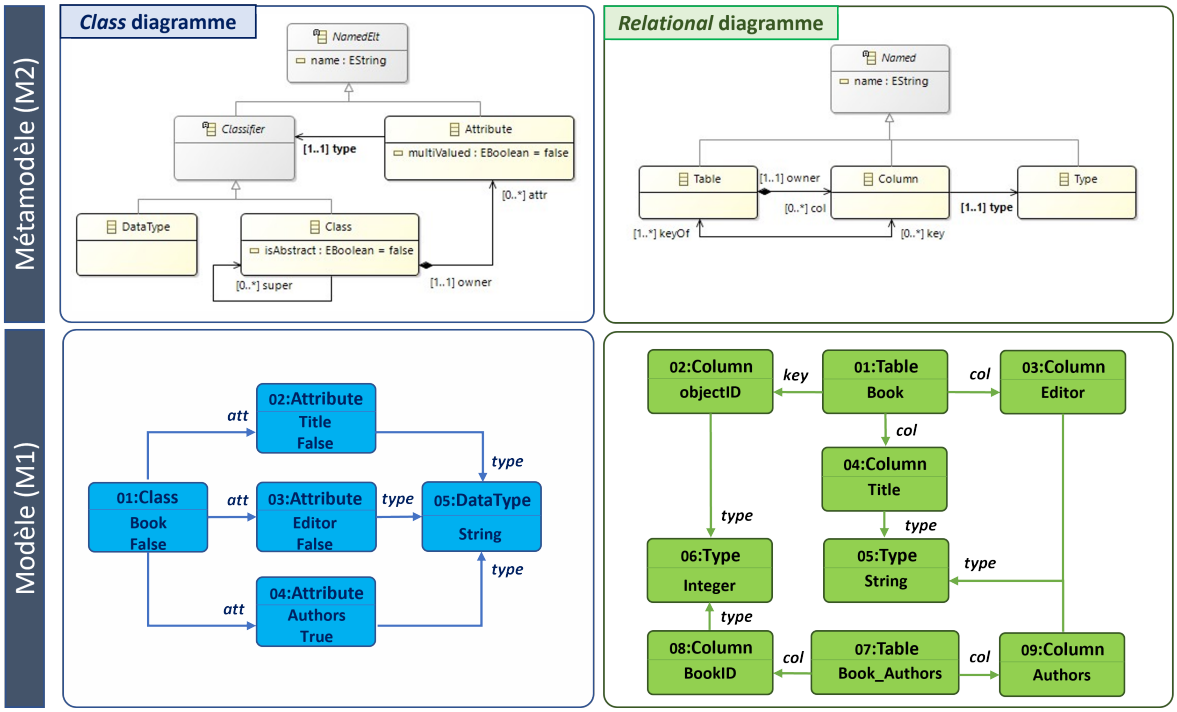


FIGURE 3.2 – Métamodèles (M2) et instances de modèles (M1) de la transformation *Class2Relational*

– *Modèle relationnel.* Le modèle relationnel est constitué d'une classe de type [*Table*] *Book* qui est elle même composée de plusieurs classes de type [*Column*] *Title*, *Editor* et *ObectID*. Dans un diagramme relationnel, chacune des lignes d'une table est identifiable par un numéro *ObectID* unique, qui est forcement associé à la classe de type [*Type*] *Integer*. Les autres classes de type [*Column*] disposent de valeurs textuelles et sont donc associées à la classe de type [*Type*] *String*.

Concernant la transformation *Families2Persons* :

– *Modèle Family.* Le modèle de *Family* est constitué d'une classe de type [*Family*] *Stark* qui est elle même composée de plusieurs classes de type [*Member*], telles que, *Johanna* liée par une association de type *mother*, *Twin* liée par une association de type *father*, *Cersi* liée par une association de type *daughter* et *Jaime* et *Tyrion* liées par une association de type *son*.

– *Modèle Persons.* Le modèle *Persons* est tout simplement constitué des classes de type [*Man*] et [*Woman*].

Avant de présenter l'approche d'apprentissage par renforcement des règles de transformation, il est important de détailler et de comprendre les différents patterns de transformation qui peuvent intervenir. Chacune des règles présentées dans les tableaux précédents, fait intervenir différents patterns de transformation. Ces patterns sont appelés caractéristiques des transformations de modèles et permettent de mettre en évidence les différents conflits sémantiques qui peuvent intervenir lors d'une transformation de modèles.

N°	***Règles en pseudo code***
R1	SI $[Class]$ ALORS CRÉATION CLASSE $[Table]$ AVEC ATTRIBUT $[Table].name = [Class].name$ CRÉATION CLASSE $[Column]$ AVEC ATTRIBUT $[Column].name = objectID$ CRÉATION RELATION $[Table] - col \rightarrow [Column]$
R2	SI $[DataType]$ ALORS CRÉATION CLASSE $[Type]$ AVEC ATTRIBUT $[Type].name = [DataType].name$
R3	SI $[Attribute].multiValued = False$ ALORS CRÉATION CLASSE $[Column]$ AVEC ATTRIBUT $[Column].name = [Attribute].name$ CRÉATION RELATION $[Column] - owner \rightarrow [Table]$ CRÉATION RELATION $[Column] - type \rightarrow [Type]$
R4	SI $[Attribute].multiValued = True$ ALORS CRÉATION CLASSE $[Table]$ AVEC ATTRIBUT $[Table].name = [Table].name + [Attribute].name$ CRÉATION CLASSE $[Column]$ AVEC ATTRIBUT $[Column].name = [Attribute].name$ CRÉATION CLASSE $[Column]$ AVEC ATTRIBUT $[Column].name = [Attribute].name +' ID'$ CRÉATION RELATION $[Table] - col \rightarrow [Column]$ x 2

TABLE 3.1 – Règles de transformation pour *Class2Relational*

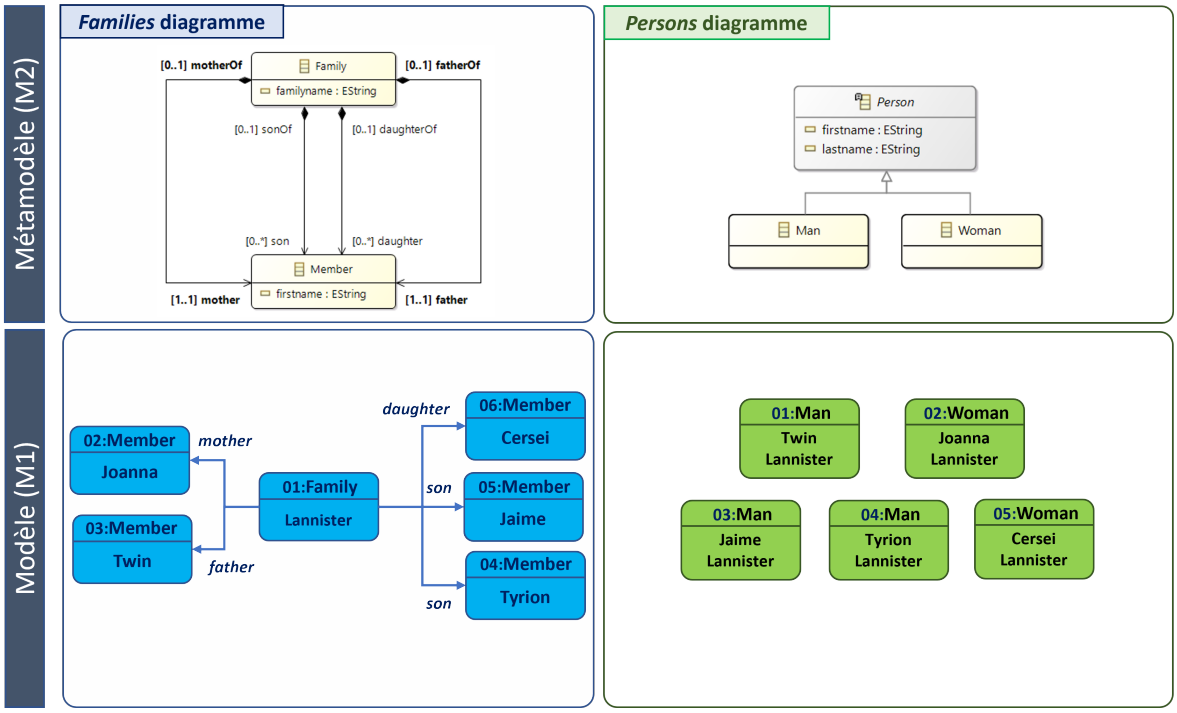


FIGURE 3.3 – Métamodèles (M2) et instances de modèles (M1) de la transformation *Families2Persons*

N°	***Règles en pseudo code***
R1	SI $[Member] - sonOf \rightarrow [Family]$ ALORS CRÉATION CLASSE $[Man]$ AVEC ATTRIBUT $[Man].firstname = [Member].firstname$ ATTRIBUT $[Man].lastname = [Family].familyname$
R2	SI $[Member] - daughterOf \rightarrow [Family]$ ALORS CRÉATION CLASSE $[Woman]$ AVEC ATTRIBUT $[Woman].firstname = [Member].firstname$ ATTRIBUT $[Woman].lastname = [Family].familyname$
R3	SI $[Member] - fatherOf \rightarrow [Family]$ ALORS CRÉATION CLASSE $[Man]$ AVEC ATTRIBUT $[Man].firstname = [Member].firstname$ ATTRIBUT $[Man].lastname = [Family].familyname$
R4	SI $[Member] - motherOf \rightarrow [Family]$ ALORS CRÉATION CLASSE $[Woman]$ AVEC ATTRIBUT $[Woman].firstname = [Member].firstname$ ATTRIBUT $[Woman].lastname = [Family].familyname$

TABLE 3.2 – Règles de transformation pour *Families2Persons*

3.2 Caractérisation des transformations de modèles

Les modèles de transformation sont définis par un ensemble de règles qui permet la transformation d'un pattern source, provenant d'un métamodèle source, en un pattern cible provenant d'un métamodèle cible. La caractérisation des transformations de modèles nécessite une compréhension plus approfondie des conflits à résoudre, provoqués par l'hétérogénéité des métamodèles. Caractériser les patterns de transformation, permet de comparer les transformations de modèles en fonction de leurs caractéristiques, et, *in fine*, d'évaluer leur complexité. C'est en ce sens que deux transformations de modèles qui partagent les mêmes caractéristiques, quel que soit le domaine des métamodèles, sont dites équivalentes et peuvent être comparées.

Dans leur article, Wimmer *et al.* [182] proposent une classification des types de conflits en deux grandes catégories (1) les conflits de type sémantiques ; (2) les conflits de type structuraux. Plus récemment, Melluso *et al.* [117] ont proposé, quant à eux, six types de conflits sémantiques qui interviennent au niveau : (1) du champs lexical du domaine ; (2) du schéma, i.e. du métamodèle ; (3) de la granularité du domaine ; (4) de la représentation des concepts ; (5) de la consistance des données ; (6) de la linguistique utilisée. Chen *et al.* [38] ajouteront que les conflits sémantiques sont également dus à la versatilité des données, puisqu'elles sont généralement biaisées, incohérentes, approximatives et incomplètes.

Pour synthétiser les précédents travaux énoncés, dans ce manuscrit, les conflits sémantiques seront classés sous forme de conflits (1) structuraux ; (2) terminologiques et syntaxiques.

3.2.1 Conflits structuraux

Selon les métamodèles (M2), une même information peut être représentée par différents concepts. Par exemple, deux classes liées par une relation dans un métamodèle peuvent être représentées par une et une seule classe dans un autre métamodèle (et inversement). Ainsi, un certain nombre de concepts sont utilisés pour exprimer la sémantique d'un seul et unique concept équivalent. Bien que ces deux structures de modélisation soient différentes, elles peuvent tout de même véhiculer la même information. Les conflits structuraux peuvent être catégorisés selon deux perspectives :

- Le *point de vue interne* dénote les attributs qui composent la classe. Ainsi, un conflit de structure interne se traduit par deux concepts sémantiquement similaires qui ne disposent pas, en totalité ou en partie, des mêmes attributs.
- Concernant *le point de vue externe*, il s'agit de porter son attention sur les relations et les classes adjacentes de la classe étudiée. Un conflit structurel externe implique que deux représentations sémantiquement similaires ne font pas nécessairement intervenir le même nombre de concepts, et peuvent être entourées par des structures adjacentes (i.e. des relations et des classes voisines) non-isomorphiques.

Les différents patterns de transformation suivent les cardinalités suivantes :

- Pattern $[1-1]$: un concept du métamodèle source est transformé en un concept du métamodèle cible. Il s'agit généralement de patterns faisant intervenir des conflits de structure interne, et

des conflits de type terminologiques.
- Pattern $[n-1]$: plusieurs concepts du métamodèle source sont transformés en un unique concept du métamodèle cible. Inversement pour le pattern $[1-n]$, un concept du métamodèle source est transformé en plusieurs concepts du métamodèle cible.
- Pattern $[n-m]$: plusieurs concepts du métamodèle source sont transformés en plusieurs concepts du métamodèle cible.

D'autre part, certains patterns peuvent être soumis à des conditions pouvant orienter la transformation d'un concept ou d'un groupe de concepts. Nous pouvons alors émettre l'hypothèse que la transformation est guidée par des caractéristiques dites propres aux concepts. Par exemple, la transformation d'un concept peut être orientée selon le type de l'association qu'elle entretient avec l'un de ses concepts voisins, et ce, quelle que soit la profondeur du voisinage. Par exemple, dans le cas de la transformation *Families2Persons*, la transformation de la classe de type [*Member*] dépend de l'association qu'elle entretient avec la classe de type [*Family*]. Dans ce cas, la transformation est dite guidée par une condition placée sur la relation qui lie les deux classes. Trois types de conditions ont été identifiées : les conditions sur les relations, la valeur des attributs et les classes adjacentes (voir Tableau 3.3)

Symbole	***Types de condition***	***Description***
σ_r	Condition sur la relation	La transformation d'une classe peut être conditionnée par une ou plusieurs de ses relations (ex : les relations *daughterOf* et *motherOf* orientent la transformation de la classe $Member$ vers la classe $Woman$ dans la transformation *Families2Persons* (voir Figure 3.3)).
σ_a	Condition sur la valeur d'un attribut de la classe	La transformation d'une classe peut être conditionnée par la valeur d'un ou plusieurs de ses attributs (ex : attribut *multivalued* dans la transformation *Class2Relational* (voir Figure 3.2)).
σ_c	Condition sur les classes adjacentes d'ordre i	La transformation d'une classe peut être conditionnée par une ou plusieurs de ses classes voisines (type de classe ou bien valeur spécifique d'un attribut)

TABLE 3.3 – Conditions de transformation

Par ailleurs, des conflits structuraux peuvent également intervenir lorsque deux attributs n'ont pas la même typologie de données. Par exemple, un attribut encodé au format *Integer* dans le métamodèle source peut être encodé au format *Double* dans le métamodèle cible.

3.2.2 Conflits terminologiques et syntaxiques

Un même concept peut être représenté par un vocabulaire très varié, qui dépend en partie du domaine auquel appartient le métamodèle. Lorsqu'un métamodèle est édité, la terminologie de ses éléments (i.e. noms des classes, des attributs et des relations) dépend également de l'expert qui la définit selon des choix subjectifs, ou en respectant des conventions de nommage comme le *Camel Case* par exemple. De cette façon, une même classe, un même attribut ou encore une même relation peuvent être nommés différemment selon les métamodèles. Cette hétérogénéité se traduit par des conflits terminologiques et syntaxiques.

D'une part, au niveau métamodèle (M2), les conflits terminologiques, lors du nommage des éléments, sont également provoqués par l'utilisation de synonymes, d'antonymes ou encore par des granularités différentes lors de la modélisation des données, qui font intervenir des hyponymes et des hyperonymes. Des conflits syntaxiques peuvent également intervenir lors de l'utilisation d'abréviations et de conventions d'écriture. D'autre part, au niveau modèle (M1), la valeur des attributs peut engendrer des conflits sémantiques dans le cas d'opérations mathématiques (conversions des devises, changement d'unités de valeur), ou encore lors d'approximations d'une valeur numérique (arrondies). Des conflits syntaxiques peuvent être induits dans le cas d'une concaténation de valeurs (deux attributs forment un unique attribut) et de formats de données différents (conversion d'une date au format anglais). Enfin, notons que les données industrielles contenues dans les modèles sont initialement saisies par des humains. Il n'est pas à exclure que de potentielles erreurs impliquent d'éventuelles incohérences dans les données.

3.2.3 Classification des conflits sémantiques

La classification des conflits sémantiques proposée dans ce manuscrit n'est pas exhaustive et mérite d'être complétée par de futures expériences. Cette classification devrait servir de base commune pour caractériser les patterns de transformation qui interviennent lors d'une transformation de modèle. La Figure 3.4 montre les différents types de conflits qui ont été identifiés, à savoir les caractéristiques qui définissent un modèle de transformation. Deux grilles de notation ont été créées pour caractériser un modèle de transformation en termes de caractéristiques terminologiques (voir Tableau 3.4) et structurelles (voir Tableau 3.5). Il est important de noter que ces caractéristiques peuvent être combinées pour créer des modèles de transformation encore plus complexes. Cela signifie qu'une règle de transformation peut contenir plusieurs patterns de transformation. De ce fait, lors d'un conflit structurel, un conflit terminologique peut exister en parallèle. De la même façon, un pattern de transformation de type $[n-m]$ englobe par défaut les pattern $[1-1]$, $[1-n]$, $[n-1]$. Cette inclusion a pour conséquence directe la création de patterns de transformation très variés, qui seraient difficiles à répertorier en intégralité.

Éléments	***Types de conflits***	***id***
Éléments du métamodèle (M2)	**Convention d'écriture** *(ex : CamelCase)*	(e1)
	Abréviation	(e2)
	Synonyme	(e3)
	Granularité *(ex : hyponymes, hyperonymes)*	(e4)
	Concaténation	(e5)
	Champs lexical différent/ Linguistique *(ex : différentes langues)*	(e6)
Type	**Format des données**	(t1)
Valeur de l'attribut (M1)	**Précision des données**	(d1)
	Abréviation et convention d'écriture	(d2)
	Texte descriptif	(d3)
	Opération mathématique	(d4)
	Concaténation de valeurs	(d5)
	Unité des données *(ex : conversion)*	(d6)
	Versatilité des données	(d7)

TABLE 3.4 – Caractéristiques terminologiques et syntaxiques

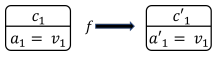
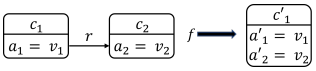
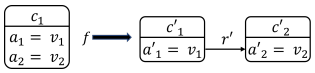
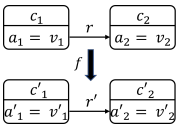
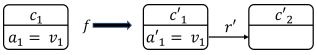
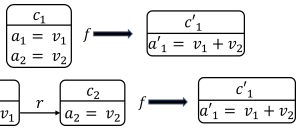
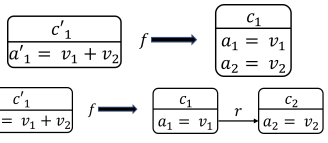
Éléments	***Types de conflits***	***id***	***Pattern***	***Pattern source et cible***
Classes	**Types des classes différents**	(c1)	$[1-1]$	
	Concaténation de classes	(c2)	$[n-1]$	
	Division d'une classe	(c3)	$[1-n]$	
	Multiplicité	(c4)	$[n-m]$	
	Création d'une dépendance	(c5) (c6)	$[1-n]$ $[n-m]$	
Attributes	**Concaténation d'attributs**	(a1) (a2)	$[1-1]$ $[n-1]$	
	Division d'attribut	(a3) (a4)	$[1-1]$ $[1-n]$	
	Composition des classes	(a5)	$[1-1]$	-

TABLE 3.5 – Caractéristiques structurelles

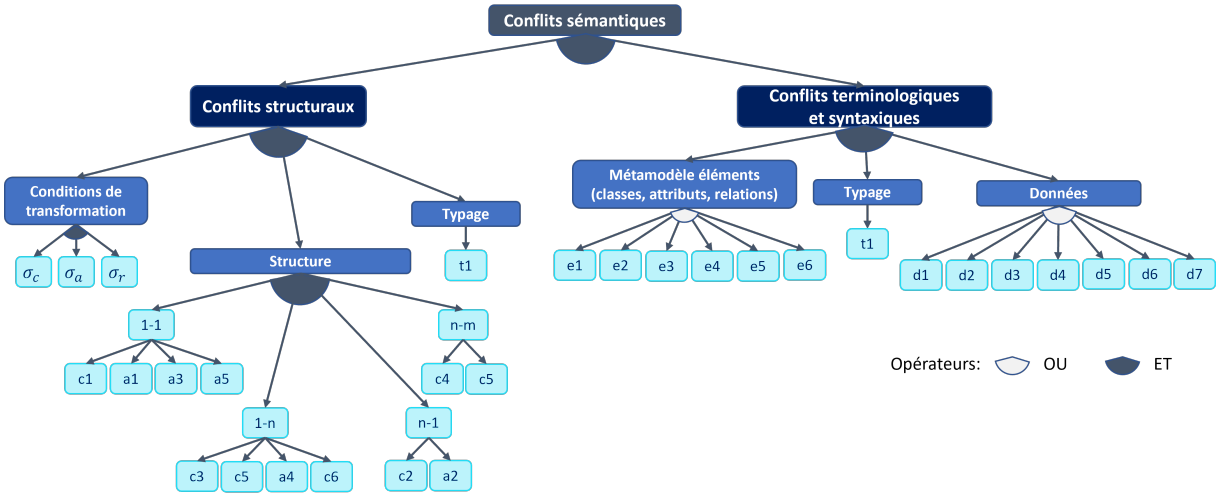


FIGURE 3.4 – Classification des conflits sémantiques

L'objectif des caractéristiques structurelles et terminologiques identifiées est de faciliter la comparaison des différentes approches de la collection expérimentale, en introduisant un cadre de références qui répertorie les différents patterns de transformation. A terme, ce cadre de références permet de mettre en évidence les caractéristiques qui peuvent ou ne peuvent pas être résolues par les approches MTBE de la collection expérimentale. Le Tableau 3.6 recense les caractéristiques structurelles et terminologiques des jeux de données identifiés dans la littérature (où C= Classe, R= Relation, A= Attribut et V= Valeur)

	Caractéristiques structurelles							***Caractéristiques terminologiques***			
	Patterns de transformation				*Conditions*						
Transformation	$[1-1]$	$[1-n]$	$[n-1]$	$[n-m]$	σ_c	σ_a	σ_r	C	R	A	V
UML2ER [183]	(c1)	-	-	(c4)	-	-	-	(e6)	(e6)	(e6)	-
Asso2Respo [53]	-	-	-	(c4)	-	-	x	(e4)	(e4)	(e3)	-
Families2Persons	-	-	(c2)	-	-	-	x	(e4)	(e4)	(e3)	-
Class2Relational	(c1)	(c5)	-	(c4)	x	x	-	(e6)	(e6)	(e6)	(d2 ;d5)
UML2Relational [46]	(c1)	(c5)	(c2)	(c4)	x	x	-	(e6)	(e6)	(e6)	(d2 ;d5)
UML2Relational [19]	(c1)	-	(c2)	(c4 ;c6)	x	x	x	(e6)	(e6)	(e6)	(d2 ;d5)

TABLE 3.6 – Caractérisation des jeux de données de validation de la collection expérimentale

Les caractérisations des transformations *Class2Relational* et *Families2Persons* sont détaillées dans la section qui suit.

3.2.4 Caractérisation des transformations *Class2Relational* et *Families2Persons*

Reprenons les transformations *Class2Relational* et *Families2Persons* présentées en Section 3.1 et appliquons les grilles de caractérisation des Tableaux 3.4 et 3.5 aux règles de transformation qui les relient.

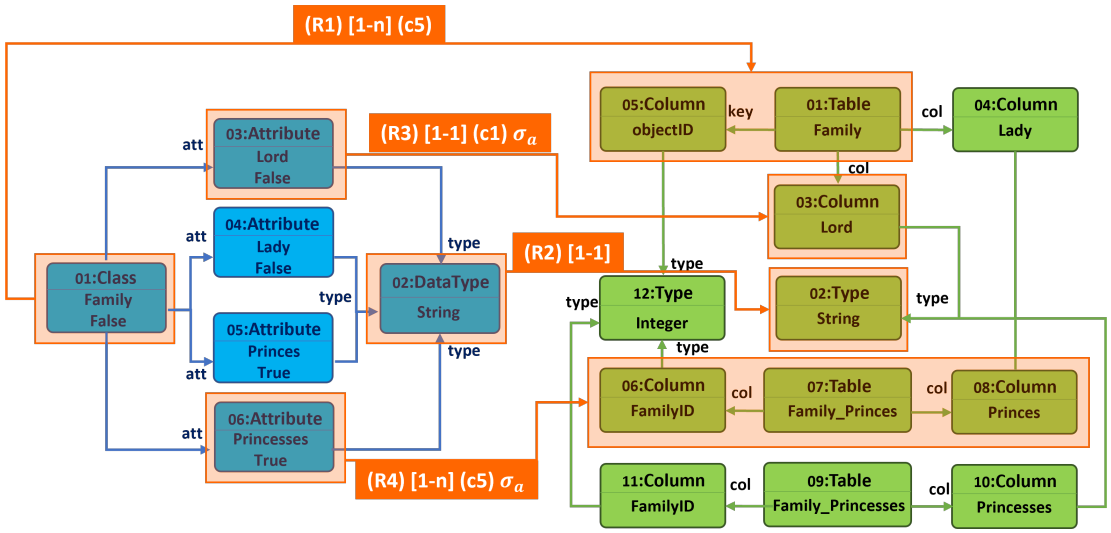


FIGURE 3.5 – Caractérisation de la transformation *Class2Relational*

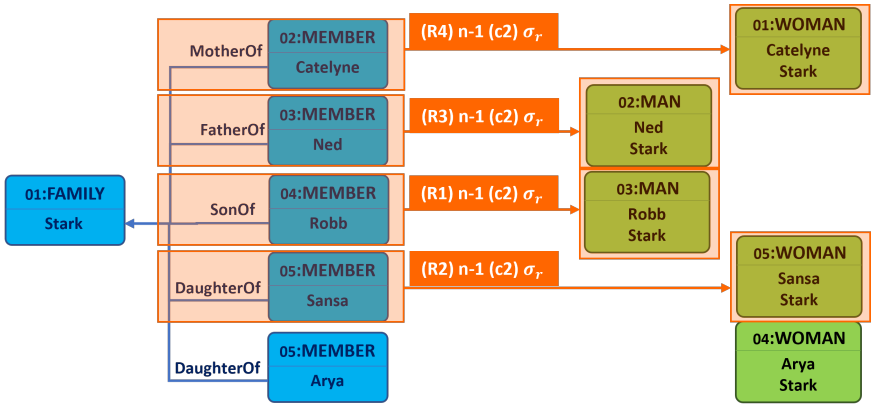


FIGURE 3.6 – Caractérisation de la transformation *Families2Persons*

Les Figures 3.5 et 3.6 expriment de manière graphique comment les règles de transformation sont caractérisées. Prenons pour exemple les règles (R1) des deux transformations. Concernant la transformation *Class2Relational*, le pattern source est constitué d'une classe ($id = 01$), tandis que le pattern cible est constitué de deux classes ($id = 01$ et $id = 05$). Ce pattern de transformation suit la cardinalité $[1 - n]$ et est considéré comme un pattern de type (c5), car la classe de type [*Table*] générée entraîne la création d'une dépendance, à savoir la création d'une classe de type [*Column*]. Notons que les règles (R1), (R3) et (R2) forment, ensemble, un pattern de transformation de type

$[n - m]$. Concernant la transformation *Families2Persons*, le pattern source est constitué de deux classes ($id = 01$ et $id = 04$), tandis que le pattern cible est constitué d'une seule classe ($id = 03$). Ce pattern de transformation suit la cardinalité $[n - 1]$ et est considéré comme un pattern de type (c2), car les classes de type [*Family*] et [*Member*] sont concaténées au sein d'une unique classe de type [*Man*].

La caractérisation des modèles de transformation permet de mettre en évidence les caractéristiques importantes à extraire des modèles, pour obtenir une représentation suffisamment détaillée des classes pour favoriser leur alignement et la dérivation des règles de transformation. Ainsi, concernant la transformation *Class2Relational*, il est nécessaire d'intégrer la structure interne de la classe étudiée, à savoir, la valeur de ses attributs, à sa représentation. Étant donné que la valeur de l'attribut *multiValued* conditionne la transformation de la classe de type [*Attribute*], il est nécessaire de prendre sa valeur *True* ou *False* en considération (condition de type σ_a). Concernant la transformation *Families2Persons*, la structure externe de la classe étudiée, à savoir, ses relations sortantes et les classes qui l'entourent, doit être considérée et intégrée à sa représentation. La relation entre la classe de type $[Member]$ et la classe de type $[Family]$ conditionne le résultat de leur transformation (condition de type σ_r).

Ces deux exemples montrent tout l'intérêt de caractériser les modèles de transformation : (1) La caractérisation permet de mettre en évidence les patterns de transformation qui interviennent dans une transformation de modèle ; (2) La caractérisation permet d'identifier les caractéristiques des classes importantes à considérer pour garantir un apprentissage des règles de transformation efficace ; (3) La caractérisation des modèles de transformation permet de comparer n'importe quelle transformation de modèles sur la base de leurs caractéristiques structurelles et terminologiques.

3.3 Apprentissage par renforcement des modèles de transformation

Pour pallier le manque d'automatisation dans la construction des transformations de modèles, une approche par apprentissage automatique des modèles de transformation est proposée. Pour éviter la traditionnelle masse de modèles requise par le corpus d'exemples des approches MTBE, les méthodes d'apprentissage par renforcement seront appliquées au problème d'apprentissage des modèles de transformation [29]. L'objectif est d'utiliser les techniques d'apprentissage Q [178], pour dériver les liens qui connectent les concepts d'un métamodèle source aux concepts d'un métamodèle cible, afin de générer automatiquement un modèle de transformation indépendant de tout langage de transformation spécifique, dans le but de garantir sa généricité. L'agent sera en charge de conjointement aligner les classes (et plus particulièrement leurs attributs), et de dériver les règles de transformation qui les relient. Pour répondre aux exigences de l'interopérabilité dynamique de type *plug and play*, la fonction de transformation apprise doit être réutilisable et applicable automatiquement, indépendamment des modèles à transformer.

Par conséquent, l'approche proposée est divisée en deux sections distinctes : (1) l'apprentissage des règles de transformation (voir Section 3.3) ; (2) la traduction, ou transformation automatique d'un modèle en réutilisant les règles de transformation apprises pendant la phase d'apprentissage (voir Section 3.5).

La phase d'apprentissage consiste à déduire les relations structurelles et sémantiques entre les concepts de deux ou plusieurs métamodèles. A la fin de la phase d'apprentissage, une fonction de transformation réutilisable, qui encapsule toutes les règles identifiées par l'agent, est créée. Cette dernière prendra la forme d'une matrice de transformation, qui intègre l'ensemble des patterns sources (provenant du métamodèle source) et leur transformation en patterns cibles (provenant du métamodèle cible). La phase d'apprentissage peut être divisée en trois étapes, présentées dans la Figure 3.7.

- La première étape consiste à injecter les informations d'un système dans l'espace technique des modèles. L'espace technique des modèles promeut un environnement unifié, où l'ensemble des modèles répond aux spécifications du méta-métamodèle *Ecore*. Cet espace facilite l'opération de transformation de modèles ;
- La deuxième étape vise à extraire la représentation structurelle et sémantique de chaque classe (représentation aplatie des classes), à partir des modèles source et cible. Comme nous avons pu le voir lors de la synthèse de l'état de l'art précédente, plus la représentation d'une classe expose sa sémantique, plus son alignement avec d'autres classes est favorisé. Cette étape permettra, à partir des représentations aplaties des classes des modèles source et cible, d'extraire des patterns sources et cibles qui seront combinés pour construire une collection de règles de transformation exécutable par l'argent lors de la phase d'apprentissage ;
- La troisième et dernière étape consiste à dériver, à partir de l'alignement des valeurs des attributs, les règles de transformation qui permettent de transformer un pattern source en un pattern cible. L'alignement des classes et la dérivation des règles de transformation seront réalisées simultanément par l'agent.

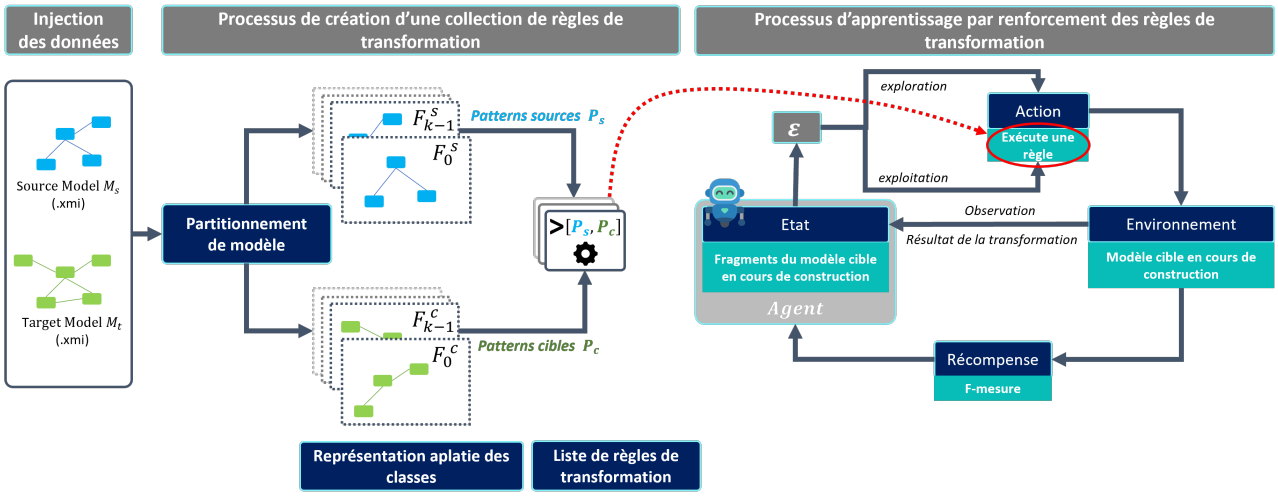


FIGURE 3.7 – Vue d'ensemble de l'approche d'apprentissage par renforcement des modèles de transformation

En résumé :

- **Données d'entrée :** Un modèle source et un modèle cible (M1).
- **Données de sortie :** Une matrice de transformation qui spécifie, pour un pattern source conforme

à une portion du métamodèle source, sa transformation en un pattern cible conforme à une portion du métamodèle cible.

3.3.1 Injection des données dans l'espace technique des modèles

Suivant les principes énoncés par l'approche MTBE, le jeu de données d'entraînement pour parfaire l'apprentissage des règles de transformation sera constitué uniquement d'une instance du modèle source et du modèle cible. Il est parfois plus simple d'identifier des correspondances entre les attributs (et par extension entre les classes) en se basant sur la similarité de leur valeur, plutôt que d'exploiter les noms des concepts eux-mêmes, qui sont souvent syntaxiquement très différents (type, i.e. nom des classes, des relations et des attributs) [8]. L'approche proposée sera donc basée sur une pair d'instance de modèle donnée à titre d'exemple, i.e. une instance pour le modèle source et une instance pour le modèle cible (M1), où, l'instance du modèle cible retranscrit comment les éléments du modèle source sont transformés selon la syntaxe du métamodèle cible. A l'aide des principes de la rétro-ingénierie, l'analyse des deux instances de modèle permet d'identifier la structure et la terminologie employées par les métamodèles source et cible, et donc, de reconstituer les deux métamodèles. Par conséquent, les règles de transformation seront dérivées entre les éléments des deux métamodèles (M2), en se basant sur l'alignement des éléments présents dans les instances de modèle (M1).

Pour illustrer l'approche proposée, la transformation bien connue *Class2Relational* présentée en Section 3.1.2, sera utilisée. Les métamodèles et modèles utilisés ont été construits à partir de l'espace technique des modèles d'*Eclipse Modeling Framework*[2] (EMF) et décrits en suivant le formalisme du méta-métamodèle *Ecore*. *Ecore* promeut un standard de modélisation unifié, qui permet de définir un cadre de modélisation commun, quel que soit le domaine d'intérêt. Ainsi, tout métamodèle (M2) est dit conforme au méta-métamodèle *Ecore* (M3).

Dans l'espace technique des modèles, les instances de modèle sont au format XMI [181] (*XML Metadata Interchange*). Le format XMI est une norme de l'OMG basée sur la structure d'un fichier XML. Il permet l'échange de modèles de données entre plusieurs systèmes dans un environnement unifié régi par le méta-métamodel *Ecore*. Ce n'est pas anodin si l'étape de transformation de modèles agit sur un modèle XMI source et produit un modèle XMI cible. L'espace technique des modèles met en avant un cadre d'interopérabilité unifié, régi par un méta-métamodel, où l'échange de données entre les systèmes est réalisé par l'intermédiaire d'un modèle de transformation.

3.3.2 Processus d'apprentissage par renforcement des règles de transformation

Définir un modèle de transformation peut être formulé comme un *Processus Décisionnel Markovien* (*Markov Decision Process MDP*), où, une séquence de décisions consiste à déterminer une suite de règles de transformation, qui permet de convertir correctement les fragments d'un modèle source en fragments d'un modèle cible. Le processus itératif qui permet de générer le modèle cible souhaité peut

2. `https://eclipse.dev/modeling/emf/`

alors être défini par l'équation suivante :

$$MDP =< S, A, T, R >$$

où S correspond à l'ensemble des états que pourra explorer l'agent, A représente l'ensemble des actions réalisables pour chaque état, T symbolise l'ensemble des transitions entre un état s_t à l'instant t et un état s_{t+1} à l'instant $t+1$ après avoir exécuté une action $a_t \in A$, et R correspond à la récompense obtenue pour avoir atteint l'état s_{t+1} après avoir réalisé une $a_t \in A$ en partant d'un état $s_t \in S$ [168].

Parmi les différents algorithmes d'apprentissage par renforcement, l'apprentissage Q a été choisi. La difficulté lorsqu'il s'agit d'appliquer les principes de l'apprentissage par renforcement est de définir une représentation suffisamment significative des états, des actions, de l'environnement dans lequel évoluera l'agent, puis d'établir une fonction de récompense en phase avec des objectifs spécifiques, qui permet de guider le comportement de l'agent durant ses expériences. La Figure 3.8 présente les éléments principaux de l'approche par renforcement, à savoir, l'environnement, les états, les actions et la récompense attribuée à l'agent.

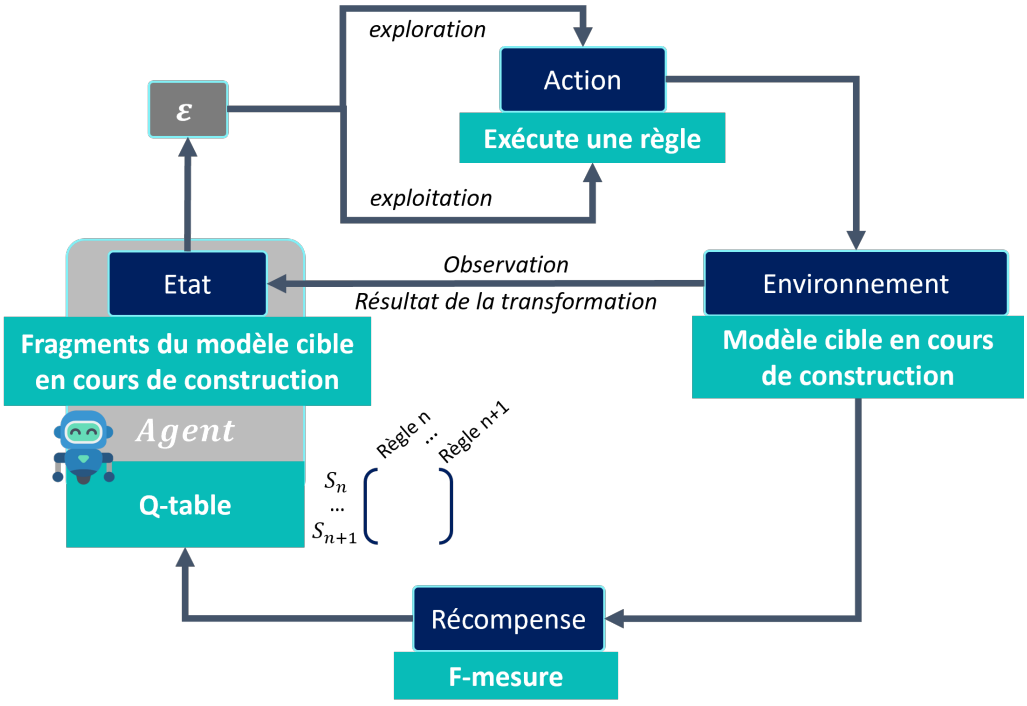


FIGURE 3.8 – Processus d'apprentissage par renforcement appliqué aux modèles de transformation

Les états. Un état $s_t \in S$ de l'environnement correspond au modèle cible intermédiaire généré par l'agent à l'instant t. S est l'ensemble des états que l'agent peut atteindre, à savoir, l'ensemble des modèles cibles intermédiaires et finaux possibles. Un modèle intermédiaire, pour un instant t, est obtenu après l'exécution d'une ou d'une succession de règles de transformation.

Les actions. Une action a_t est utilisée pour mettre à jour le modèle cible intermédiaire à un instant t. La mise à jour du modèle intermédiaire consiste à ajouter des fragments de modèle cible créés à

partir d'une règle de transformation. En d'autres termes, dans le cas d'un modèle de transformation, réaliser une action revient à sélectionner des fragments du modèle source, puis à les transformer en fragments du modèle cible (voir processus d'exécution d'une règle de transformation en Section 3.4.3). Par conséquent, A est l'ensemble des règles de transformation qui permet d'apporter des modifications au modèle intermédiaire actuel. Le processus d'initialisation d'une collection de règles de transformation est présenté en Section 3.4. Un fragment du modèle source ne peut être converti par une règle de transformation qu'une seule fois par épisode. Cela signifie que pour un état donné, en fonction des règles de transformation précédemment appliquées, l'agent dispose d'une collection filtrée de règles de transformation qui peuvent être exécutées.

Les transitions. Les modifications apportées aux modèles cibles intermédiaires à chaque instant t suivent une distribution de probabilités basée sur l'ensemble des transitions possibles pour chaque état s_t. En d'autres termes, $P(s_{t+1}|s_t, a_t) = T(s_t, a_t, s_{t+1}) \rightarrow [0, 1]$ est la probabilité que l'agent atteigne l'état s_{t+1} en partant de l'état s_t, après avoir exécuté une action a_t. En résumé, les transitions d'un processus décisionnel Markovien modélisent comment un système évolue en réponse aux actions prises par un agent. Ces transitions, associées à des récompenses, permettent à l'agent de prendre des décisions éclairées, qui permettent de maximiser le total cumulatif des récompenses pour une séquence d'actions. Dans le cas d'une transformation de modèles, une transition modélise le passage d'un modèle cible intermédiaire à l'instant t en un nouveau modèle cible intermédiaire à l'instant $t + 1$, après avoir exécuté une règle de transformation.

La récompense. $R(s_t, a_t, s_{t+1})$ est la récompense reçue par l'argent en passant de l'état s_t à l'état s_{t+1}, après avoir réalisé une action a_t. Dans le cas d'une transformation de modèles, il s'agit d'évaluer si le modèle cible intermédiaire obtenu à l'instant $t + 1$ est correct. Évaluer la composition du modèle cible intermédiaire à l'instant $t + 1$ implique de devoir mesurer l'impact que l'action a_t, à savoir la règle de transformation exécutée, a eu sur le modèle cible intermédiaire à l'instant t. Nous avons donc défini le calcul de la récompense de la façon suivante :

$$R(s_t, a_t, s_{t+1}) = F_\beta.coh$$

où le F_β correspond au score de F-mesure qui prend en considération la précision P, i.e. le ratio entre le nombre d'éléments produits attendus et le nombre total d'éléments produits, et le *recall* R, i.e. le ratio entre le nombre d'éléments produits attendus et le nombre total d'éléments attendus, tel que :

$$F_\beta = (1 + \beta^2)\frac{P.R}{\beta^2.P + R} \rightarrow [0, 1]$$

où β attribut un score d'intérêt en faveur du *recall*, ou de la précision. Par exemple, pour $\beta < 1$, l'importance de l'indicateur de *recall* sera plus fort que celui de l'indicateur de précision. Le score de F-mesure permet d'évaluer à chaque instant t si l'agent converge vers une solution optimale qui se traduit par un score de F-mesure= 1.0. Pour obtenir le score de F, l'agent compare le modèle cible souhaité, donné à titre d'exemple, avec le modèle cible intermédiaire obtenu pour un instant $t + 1$. L'apprentissage est donc guidé par le modèle cible souhaité, qui donne une indication sur la façon dont

la transformation doit se comporter. Dans le cas d'une transformation de modèle, une F-mesure= 1.0 signifie que l'ensemble des éléments du modèle cible (M1) a correctement été construit et assemblé. Par conséquent, un score de F-mesure maximal induit qu'une succession de règles de transformation a permis d'obtenir le modèle cible souhaité.

Le score *coh* est un indicateur de cohérence qui permet d'évaluer si l'action a_t qui vient d'être exécutée est cohérente avec les actions passées. L'indicateur de cohérence détermine si les modifications apportées au modèle cible intermédiaire, à l'instant t, ont affecté positivement ($coh = 1$), ou négativement ($coh = 0$), le modèle cible intermédiaire obtenu à l'instant $t+1$. En d'autres termes, les fragments générés par l'action a_t viennent modifier l'état actuel du modèle intermédiaire à l'instant t. Ces modifications se traduisent par l'ajout d'éléments de type *EClass*, par l'intégration d'éléments de type *EAttribute* au sein d'éléments de type *EClass* préexistants et par la création d'éléments de type *EReference* entre les éléments de type *EClass* préexistants à l'instant t et qui viennent d'être créés à l'instant $t+1$. Ainsi, l'indicateur de cohérence évalue la structure interne et la structure externe des classes du modèle intermédiaire obtenu à l'instant $t+1$. Un score $coh = 0$ signifie que, soit une ou plusieurs des classes est constitué des mauvais attributs, soit que les associations entre les différentes classes sont inadéquates.

Politique d'optimisation. En apprentissage Q, l'ensemble des expériences menées par l'agent est stocké dans un tableau appelée *Q-table*. Lorsque l'apprentissage débute, tous les couples états-actions de la Q-table sont initialisés à zéro. La *Q-valeur* pour chaque couple état-action est mise à jour en fonction des interactions conduites par l'agent dans l'environnement, en appliquant l'équation de *Bellman* présentée en Section3.1.1. L'apprentissage se déroule sur plusieurs épisodes définis à l'avance. L'agent suit une politique d'exploration et d'exploitation de type *ϵ-greedy* qui lui permettra de se rendre dans l'ensemble des états possibles. Cela signifie que l'agent peut à la fois explorer son environnement en choisissant des actions aléatoires avec la probabilité ϵ, et exploiter les connaissances acquises précédemment pendant son apprentissage avec la probabilité $1-\epsilon$ (soit en choisissant la valeur Q la plus élevée pour le couple état-action). Au fur et à mesure des itérations de l'algorithme, la valeur de ϵ diminue et pousse l'agent à agir de façon optimale en exploitant ses connaissances. Le Tableau 3.7 présente le déroulement de l'algorithme d'apprentissage Q.

Politique optimale. Résoudre un problème structuré sous la forme d'un MDP consiste à déterminer la politique optimale qui permet à l'agent de cumuler un maximum de récompenses lors de son évolution à travers les états de l'environnement. Agir de façon optimale revient à suivre la politique optimale π^*, où, pour un état s_t, l'action a_t qui maximise l'espérance des récompenses futures est retenue. La sélection d'une action optimale par l'agent suis l'équation :

$$\pi^*(s_t) = \underset{a_t}{argmax} Q^*(s_t, a_t)$$

Dans le cas d'une transformation de modèle, une séquence optimale d'actions revient à exécuter un ensemble de règles de transformation, qui permet de transformer un modèle source en un modèle cible correspondant. Une fois l'apprentissage terminé, une matrice de transformation M_T encapsule l'ensemble des règles de transformation trouvées par l'agent. Les lignes de la matrice M_T correspondent

	Algorithme : Processus d'apprentissage des règles de transformation
	Entrée : Modèle source M_s ; Modèle cible M_c
	Sortie : Q-table ; Matrice de transformation M_T
1	Initialisation de la Q-table : $Q(S, A)$
2	**Pour chaque** épisode i **faire**
3	**Tant que** tous les fragments du modèle source n'ont pas été transformés
4	Choisir une règle de transformation $a \in A$ depuis l'état s_t suivant la politique ϵ-*greedy*
5	Exécuter la règle de transformation a_t
6	Observer le résultat de la transformation, calculer la récompense r
7	Mettre à jour $Q_{i+1}(s_t, q_t) = Q_i(s_t, q_t) + \alpha[r_{t+1} + \gamma \underset{a_{t+1}}{max} Q_i^*(s_{t+1}, a_{t+1}) - Q_i(s_t, q_t)]$
8	Filtrer les règles de transformation possibles pour l'état s_{t+1}
9	Diminuer ϵ
10	Générer la matrice de transformation M_T

TABLE 3.7 – Processus d'apprentissage Q pour la dérivation des règles de transformation

aux différents patterns sources issus du métamodèle source, tandis que les colonnes représentent les patterns cibles issus du métamodèle cible. La matrice M_T peut être formulée de la façon suivante :

$$M_T = \begin{bmatrix} [\mathrm{P}_{s_1}, P_{c_1}] & ... & [\mathrm{P}_{s_1}, P_{c_n}] \\ ... & ... & ... \\ [\mathrm{P}_{s_n}, P_{c_1}] & ... & [\mathrm{P}_{s_n}, P_{c_n}] \end{bmatrix}$$

$$\text{où } [P_{s_n}, P_{c_n}] = \begin{cases} 1 & \text{si la règle est prouvée} \\ 0 & \text{si la règle n'a pas été démontrée} \end{cases}$$

Pour toutes nouvelles instances de modèle conforme au métamodèle source, la matrice de transformation peut être directement réutilisée dans un programme de transformation, pour obtenir le modèle cible équivalent, conforme à la syntaxe du métamodèle cible.

3.4 Processus de création des règles de transformation

Avant tout, il est important d'apporter certaines définitions essentielles, pour clarifier le processus de création d'une collection de règles de transformation exploitables par l'agent.

Règle de transformation. Par définition, une règle de transformation spécifie comment un pattern source (LHS pour *Left-Hand Side*) P_s, est transformé en un pattern cible (RHS pour *Right-Hand Side*) P_c [47].

Patterns source et cible. Un pattern, est une portion de métamodèle. Cette portion de métamodèle (M2) symbolise une structure récurrente, qui a tendance à se répéter dans une instance de modèle (M1). Un pattern est composé d'une collection d'éléments qui provient du métamodèle. Selon le formalisme du méta-métamodèle *Ecore*, ces éléments sont de type *EClass*, *EAttribute* et *EReference*. Ainsi, dans

le cas d'une transformation de modèle, une règle précise la transformation d'un groupe d'éléments, conforme à la syntaxe du métamodèle source, en un groupe d'éléments conforme à la syntaxe du métamodèle cible.

Fragments de modèle. Par analogie, un pattern est semblable à un patron de couture que l'on suit pour créer des motifs spécifiques. Dans le cas d'une transformation de modèle, le patron en question permet d'identifier les fragments de modèle (M1) conformes au pattern défini au niveau métamodèle (M2). De cette façon, lors d'une transformation de modèle, un pattern source permet de filtrer les fragments du modèle source, qui vont être transformés selon les spécifications du pattern cible. La transformation des fragments du modèle source aboutira à la création de fragments du modèle cible conformes au pattern cible indiqué par la règle de transformation.

Établir un programme de transformation basé sur des règles repose essentiellement sur la définition de couples $[P_s, P_c]$. Il est alors indispensable d'identifier les différents patterns présents dans les modèles source et cible donnés à titre d'exemple, pour édifier une collection de règles de transformation que l'agent sera en charge de tester. Par conséquent, il est nécessaire d'extraire la structure interne et externe des classes des modèles source et cible (M1), pour faciliter la création des patterns sources et cibles qui représentent des portions des métamodèles source et cible (M2). L'identification des patterns sources et cibles est une étape obligatoire, qui permet de dériver les règles de transformation qui lient les concepts de métamodèle source aux concepts du métamodèle cible.

Pour détecter les différent patterns présents dans les modèles source et cible, les techniques de partitionnement de modèle seront exploitées dans la section suivante.

3.4.1 Partitionnement des modèles source et cible

Déterminer les patterns récurrents présents dans un modèle est une opération de division qui consiste à découper un modèle en plusieurs petits modèles. Cette approche fait écho aux *techniques de partitionnement* utilisées en *théorie des graphes*, pour diviser l'ensemble des entités d'un graphe en plusieurs sous-ensembles, ou sous-graphes, appelés "parties". Un graphe est une représentation adaptée pour décrire la structure sous-jacente d'un modèle. Une transformation de modèles est d'ailleurs très souvent formulée sous l'angle d'un problème de transformation de graphes [158]. De ce fait, la capture de la structure interne et externe des classes est inspirée de la théorie des graphes, ainsi que des récentes approches d'alignement et d'appariement des graphes, qui tentent d'exprimer une représentation suffisamment significative des entités pour favoriser leur mise en relation. Les approches d'apprentissage de représentation des entités ont montré leur efficacité, lorsqu'il s'agit de mesurer la similarité entre deux entités dans un espace vectoriel latent. Cependant, ces dernières approches d'apprentissage de représentation nécessitent un grand nombre de pré-alignements [155].

Pour palier le manque de pré-alignement, et exprimer les différentes caractéristiques de la structure des modèles source et cible, le choix a été fait de représenter chaque classe par sa *représentation aplatie* [59]. La représentation aplatie d'une classe permet de capturer les caractéristiques internes (propres, i.e. type d'attributs de la classe) et externes (héritées, i.e. type d'associations entretenues avec d'autres classes) d'une classe. En d'autres termes, le contexte de chaque classe est intégralement exprimé à

travers ses propriétés propres et héritées. Une représentation aplatie permet de différencier les classes de type similaire qui pourraient avoir un contexte différent, à savoir des valeurs d'attributs spécifiques ou encore des associations particulières avec d'autres types de classes.

Afin de formuler la représentation aplatie d'une classe, il est nécessaire au préalable d'extraire un certain nombre d'éléments issus de la structure externe et interne d'une classe. Commençons par définir un modèle selon les spécifications du méta-métamodèle *Ecore*. Un modèle (M1) peut être défini par un ensemble de classes de type *EClass* et d'associations de type *EReference*. Chaque classe peut contenir un ou plusieurs attributs de type *EAttribute*, qui sont typés en fonction du format des données (numérique, caractère, date...). On peut alors définir la représentation M d'un modèle par l'équation $M = (C, R, A, V)$, où C est un ensemble de classes (définies par leur type, i.e. un nom), R un ensemble de références, orientées ou non, (définies par leur type, i.e. un nom) qui associent les classes entre elles, A un ensemble d'attributs qui composent les classes et V l'ensemble des valeurs des attributs. A partir de cette définition, nous pouvons maintenant présenter les éléments qui caractériseront la structure interne et externe des classes, afin d'édifier les fameuses classes aplaties qui permettront de partitionner les modèles source et cible et d'obtenir, *in fine*, les patterns sources et cibles.

La **structure interne** d'une classe est caractérisée par les attributs qu'elle contient ainsi que les valeurs qui leur sont attribuées. Deux éléments sont essentiels à extraire :

1. *Doublets en attribut.* $D^a = \{(c, a)|c \in C, a \in A\}$. Les doublets en attribut permettent de capturer, au niveau métamodèle, la structure interne d'une classe, à savoir les types d'attributs qui la composent. Un doublet en attribut est de la forme $< EClass, EAttribute >$. Suivant cette définition, la classe de type $[Class]$ a pour doublet en attribut $< Class, name >$.
2. *Triplets en attribut.* $T^a = \{id(c, a, v)|c \in C, a \in A, v \in V\}$, où id correspond à l'identifiant unique associé à chaque classe. Dans le cas où une classe détiendrait plusieurs attributs, l'id permet d'associer chacun des triplets en attribut à une unique classe. Les triplets en attribut capturent la valeur de chacun des attributs. Un triplet en attribut est de la forme $< EClass, EAttribute, valeur >$. Suivant cette définition, la classe de type $[Class]$ a pour triplet en attribut $< Class, name, Family >$.

La **structure externe** d'une classe se traduit quant-à elle par les associations qu'elle entretient avec les classes qui l'entourent.

3. *Triplets en relation.* $T^r = \{(h, r, t)|h, t \in C, r \in R\}$ avec r la référence orientée depuis la classe h ($head$) vers la classe t ($tail$). Les triplets en relation capturent les propriétés hérités, i.e. issues des associaions avec d'autres classes. Ces triplets sont déterminés à partir de la matrice d'adjacence AD. La matrice exprime les interactions entre les classes, à savoir les relations entre elles. Un triplet en relation est de la forme $< EClass, EReference, EClass >$. De ce fait la classe de type $[Class]$ détient le triplet en relation $< Class, attr, Attribute >$.

Le partitionnent d'un modèle en classes aplaties suit une approche heuristique définie par une succession d'étapes appliquées à l'ensemble des classes d'un modèle :

(1) Récupération des doublets en attribut D^a_c pour la classe c étudiée.

(2) Récupération des triplets en relation T^r_c pour la classe c étudiée. Les classes qui ne disposent pas de triplets en relation génèrent des classes aplaties qui ne disposent pas de caractéristiques

héritées.

L'opération de partitionnement d'un modèle, à savoir, la division d'un modèle en classes aplaties, peut alors être exprimée de la façon suivante :

Un modèle M est partitionné en k parties, qui dépendent du nombre de classes aplaties F identifiées, tel que $P = [F_0, ..., F_{k-1}]$,où une classe aplatie est formulée par l'équation $F = (c, D_c^a, T_c^r$ avec c le type, i.e. le nom de la classe dont est issue la classe aplatie. Les Figures 3.9 et 3.11 présentent les classes aplaties obtenues après l'opération de partitionnement des modèles source et cible.

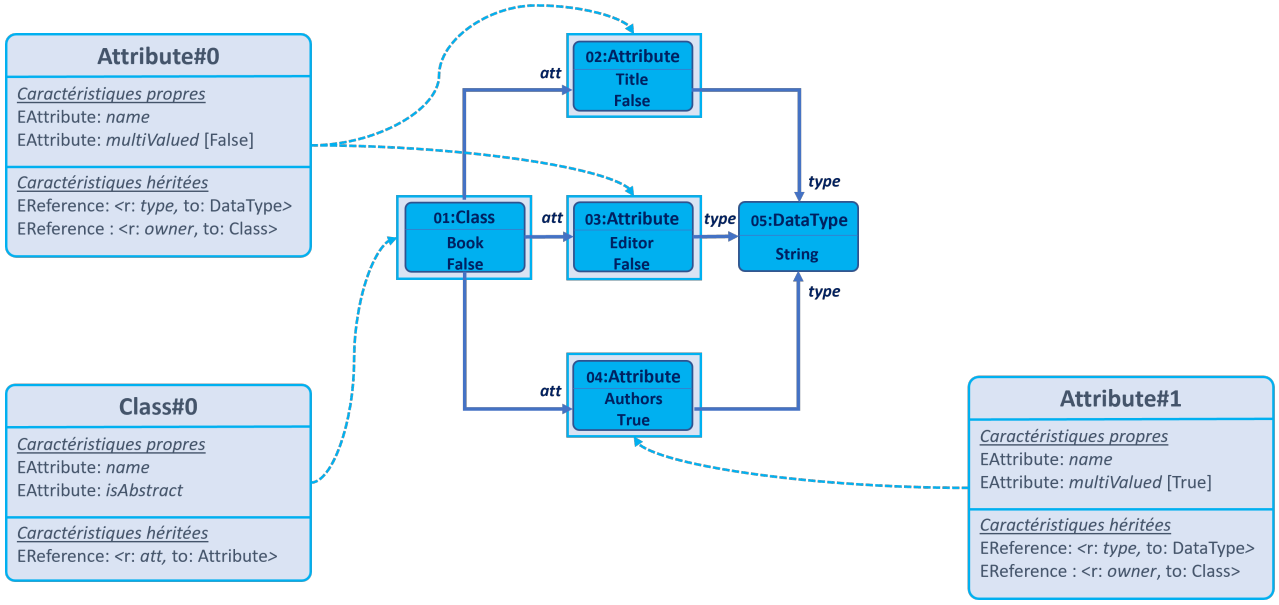
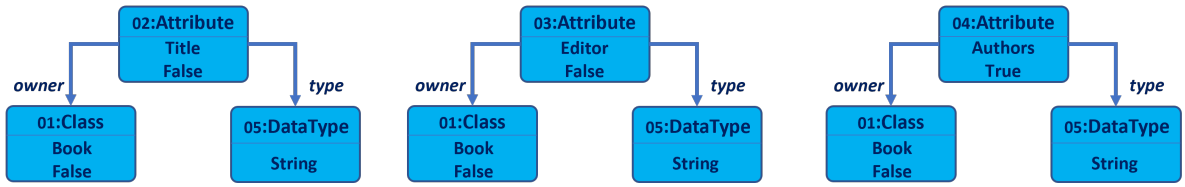


FIGURE 3.9 – Classes aplaties obtenues pour le modèle de classes

La création des classes aplaties peut s'apparenter à une activité de *clustering*, qui permet de regrouper les classes similaires d'un modèle (M1) en fonction de leur structure (interne et externe). L'objectif d'un algorithme de *clustering* est de regrouper les classes qui ont des caractéristiques communes. Dans le cas d'une opération de partitionnement de modèles, où le modèle est découpé en plusieurs classes aplaties, il s'agit d'identifier et de regrouper les classes d'un même type, qui comportent des valeurs d'attributs identiques et des associations avec d'autres classes similaires. Par exemple, selon la Figure 3.9, les classes de type $[Attribute]$ ayant pour id 02 et 03 ont le même contexte. Du point de vue de leur structure externe, ces deux classes sont constituées des mêmes associations, à savoir les triplets en relation $< Attribute, type, DataType >$ et $< Attribute, owner, Class >$. Concernant leur structure interne, en plus de contenir les mêmes attributs, à savoir les $EAttribute < name >$ et $< multiValued >$, ces deux classes comportent les mêmes valeurs spécifiques pour l'attribut $< multiValued >$, identifiables par les triplets en attribut $< Attribute, multiValued, False >$. Par conséquent, les classes $id = [03, 04]$ ont donc été classées dans le même groupe nommé Attribute#0. Dans les classes aplaties, les valeurs spécifiques des attributs, qui ont permis d'édifier les classes aplaties, sont notés entre crochets à côté de l'élément *EAttribute* ou *EClass* concerné.



(a) Classes issues de la représentation aplatie Attribute#0 (b) Classe aplatie Attribute#1

FIGURE 3.10 – Fragments du modèle de classes conformes aux classes aplaties Attribute#0 et #1

La Figure 3.10 illustre clairement l'opération de classification des fragments du modèle source selon les classes aplaties générées. Ainsi, l'opération de partitionnement, s'assimile à une opération de type *clustering* qui permet de classer des fragments du modèle source, ou sous-modèles, en fonction de leur structure interne et externe. Selon la théorie des graphes, capturer la structure externe d'une classe revient à capturer sa structure adjacente d'ordre-k, i.e. les classes voisines avec lesquelles une relation est entretenue. Dans le cas présent, la structure adjacente capturée par la classe aplatie est d'ordre 1.

Concernant les classes aplaties générées par le modèle relationnel (voir Figure 3.11), nous pouvons constater que les valeurs des attributs du modèle cible, inconnues du modèle source, sont substituées par le symbole "?", à savoir, les valeurs *Book_Authors, BookID, Integer, ObjectID*. Typiquement, les *EClass* ayant pour $id = [02, 06, 07, 08]$ sont des éléments créés à partir d'une règle de transformation. Il est donc normal de les masquer. L'agent sera en charge d'inférer les règles de transformation qui permettront la création de ces fragments de modèle. Enfin, il est important de noter que les valeurs "?" agissent comme des valeurs spécifiques à la classe. Par conséquent, bien que les classes aplaties Column#1 et Column#2 aient la même structure interne et externe, les valeurs de leurs attributs les distinguent. Les classes ayant pour id 03 et 02 sont donc classées dans deux groupes différents.

Il est essentiel de souligner que les représentations aplaties générées proviennent d'une approche heuristique. Cela signifie que le partitionnement des modèles peut suivre des politiques toutes aussi différentes. Par exemple, pour certaines transformations, il serait judicieux d'inclure le contexte à gauche des classes, à savoir les associations orientées vers la classe étudiée. Intégrer l'ordre $k > 1$ des classes pourrait être également une solution envisageable, lorsqu'il s'agit de considérer des patterns plus étendus, qui nécessitent une compréhension plus profonde des associations entre les classes. Pour le moment, nous avons fait le choix de construire la représentation aplatie d'une classe en fonction des associations sortantes (contexte à droite) que les classes détiennent, de sorte à limiter le nombre de classes aplaties générées. Il est prévu d'enrichir cette représentation dans de futures expérimentations.

Ne perdons pas de vue que le processus de partitionnement des modèles sources et cibles, en fonction des représentations aplaties des classes, n'est qu'une étape intermédiaire pour caractériser le contexte de chaque classe, puis inférer les patterns sources et cibles qui serviront à construire des règles de transformation.

Les classes aplaties obtenues pour la transformation *Families2Persons* sont présentées en Annexe A.

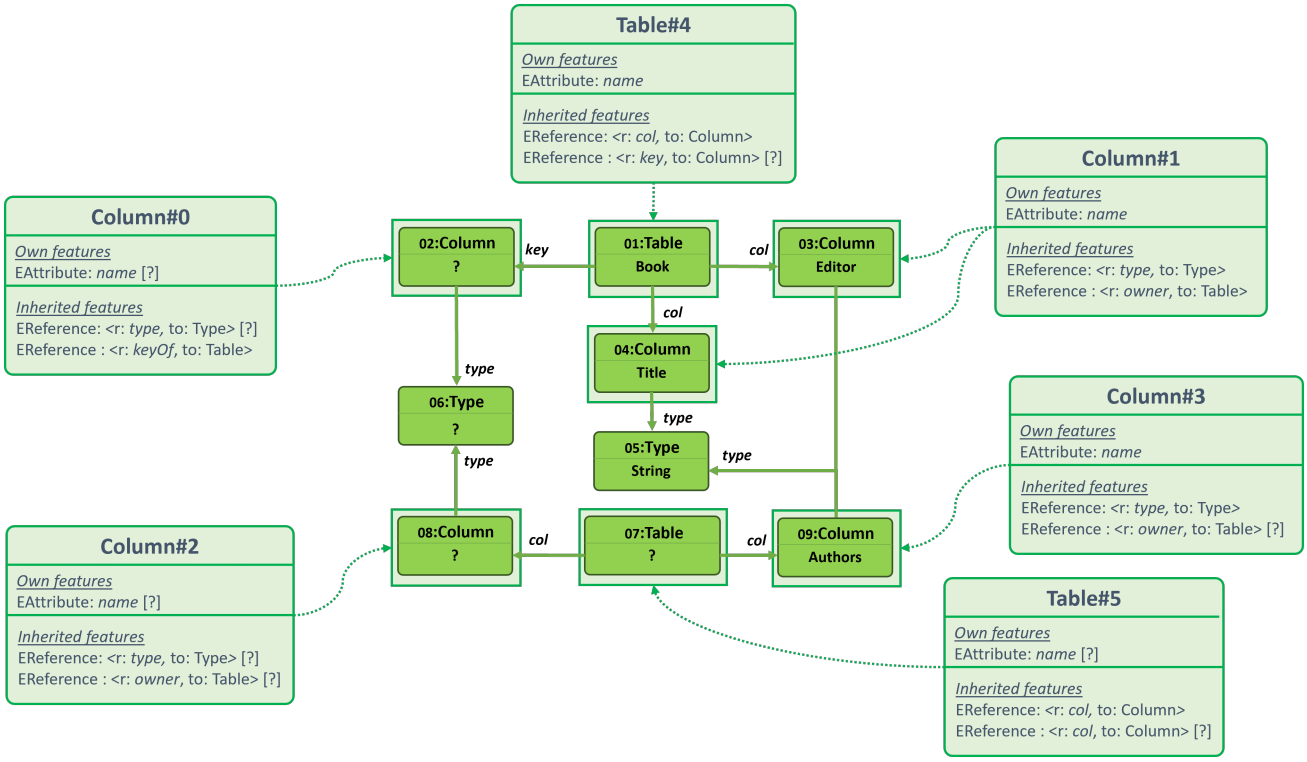


FIGURE 3.11 – Classes aplaties obtenues pour le modèle relationnel

3.4.2 Création des patterns sources et cibles

A présent que les modèles source et cible ont été partitionnés en fonction du nombre de classes aplaties qu'ils détenaient, il est temps maintenant de construire les patterns sources et cibles qui vont permettre d'édifier une collection de règles de transformation, que l'agent sera en charge de tester pendant la phase d'apprentissage.

Étant donné que l'approche consiste à aligner les attributs des modèles source et cible en fonction de leur valeur et du contexte dans lequel ils interviennent, il est important que le processus d'apprentissage des règles de transformation prenne en considération la représentation aplatie des classes dont sont issus les attributs. La représentation aplatie d'une classe spécifie le contexte dans lequel intervient un attribut. Un pattern peut ainsi être formulé par le vecteur à trois dimensions suivant :

$$P = [Contexte, EClass, EAttribute]$$

où la première dimension spécifie une valeur de contexte (le numéro correspondant à la classe aplatie), la seconde dimension spécifie l'élément *EClass* rattaché au contexte et la troisième et dernière dimension est propre à l'élément *EAttribute* appartenant à la *EClass*. Deux types de patterns peuvent être inférés à partir d'une classe aplatie :

– *Patterns inférés depuis les caractéristiques propres.* Les caractéristiques propres d'une classe aplatie correspondent aux éléments de type *EAttribute* de la classe principale. Ainsi, pour

chaque élément *EAttribute*, un pattern est créé. Par exemple, pour la classe aplatie Attribute#0, les patterns issus des caractéristiques propres de la classe de type $[Attribute]$ sont $[0, Attribute, name]$ et $[0, Attribute, multivalued]$.

– *Patterns inférés depuis les caractéristiques héritées.* En principe, suivant la théorie des graphes, une classe adjacente (classe reliée par un association de type *EReference*) hérite du contexte de la classe principale. Par conséquent, il est important que des patterns retranscrivent le contexte de la classe principale depuis le point de vue des classes adjacentes. Cela se traduit par la création de patterns pour chaque *EClass* reliée par une association de type *EReference* avec la classe principale qui a générée la représentation aplatie. Ainsi, pour la classe aplatie Attribute#0, les patterns issus des caractéristiques héritées de la classe de type $[Attribute]$ sont $[0, DataType, nam]e$, $[0, Class, name]$ et $[0, Class, multiValued]$.

Suivant ces deux processus de création de patterns, nous obtenons, *in fine*, un ensemble de patterns sources P_s et de patterns cibles P_c. Pour constituer des règles de transformation, l'agent sera en charge de tester les différents couples $[P_s, P_c]$ et de déterminer la succession de règles, qui permettra de transformer correctement le modèle source M_s afin d'obtenir le modèle cible M_t souhaité. L'entraînement sera ainsi guidé par le couple de modèles $[M_s, M_t]$.

3.4.3 Processus d'exécution d'une règle de transformation

Dans le cadre d'un processus d'apprentissage par renforcement, l'agent prend des décisions en choisissant parmi un ensemble d'actions possibles selon l'état atteint à l'instant t. Ces actions déterminent la manière dont l'agent interagit avec son environnement. Considérant un problème de transformation de modèles, réaliser une action revient à exécuter une règle de transformation $[P_s, P_c]$, qui permet de transformer les fragments d'un modèle source conformes à un pattern source P_s, en fragments d'un modèle cible conformes à un pattern cible P_c. Les fragments produits par la règle de transformation sont intégrés au modèle cible intermédiaire existant à l'état s_t, ce qui projette l'agent dans un état s_{t+1}.

Suivant la Figure 3.12, à l'instant t, l'agent sélectionne la règle de transformation $[0, Class, name] \rightarrow [1, Table, name]$ qui peut être définie par le pseudo-code décrit dans le Tableau 3.8

	Algorithme : Règle de transformation $[0, Class, name] \rightarrow [1, Table, name]$
	Entrée : Fragment du modèle source sous forme de triplet en attribut T_s^a **Sortie** : Fragment du modèle cible
1	**Pour chaque** *EClass* de type $[Class]$ dans le contexte Attriubte#0
2	**Construire** une *EClass* de type $[Table]$ contenant
3	*EAttribute* $[Table].name = [Class].name$

TABLE 3.8 – Pseudo-code de la règle de transformation $[0, Class, name] \rightarrow [1, Table, name]$ exécute par l'agent

Les fragments du modèle source conformes au pattern source $[0, Class, name]$ sont matérialisés par les triplets en attribut source T_s^a. Pour le pattern source choisi, l'agent filtre la liste des triplets en attribut conforme au contexte de la classe aplatie #0 pour la classe de type $[Class]$ et récupère le triplet $< Class, name, Book >$. Le résultat de la transformation, suite à la règle exécutée par l'agent (voir Tableau 3.8), correspond au fragment du modèle cible conforme au pattern cible $[1, Table, name]$. En d'autres termes, une *EClass* de type $[Table]$ est créée, dans laquelle un *EAttribute* de type *name* ayant pour valeur *Book* est intégré. Le fragment du modèle cible obtenu après transformation, matérialisé par un triplet en attribut cible T_c^a, correspondant au triplet $< Table, name, Book >$ est ensuite intégré au modèle cible intermédiaire, qui est déjà composé des fragments de modèle cible $< Column, name, Title >$ et $< Column, name, Editor >$.

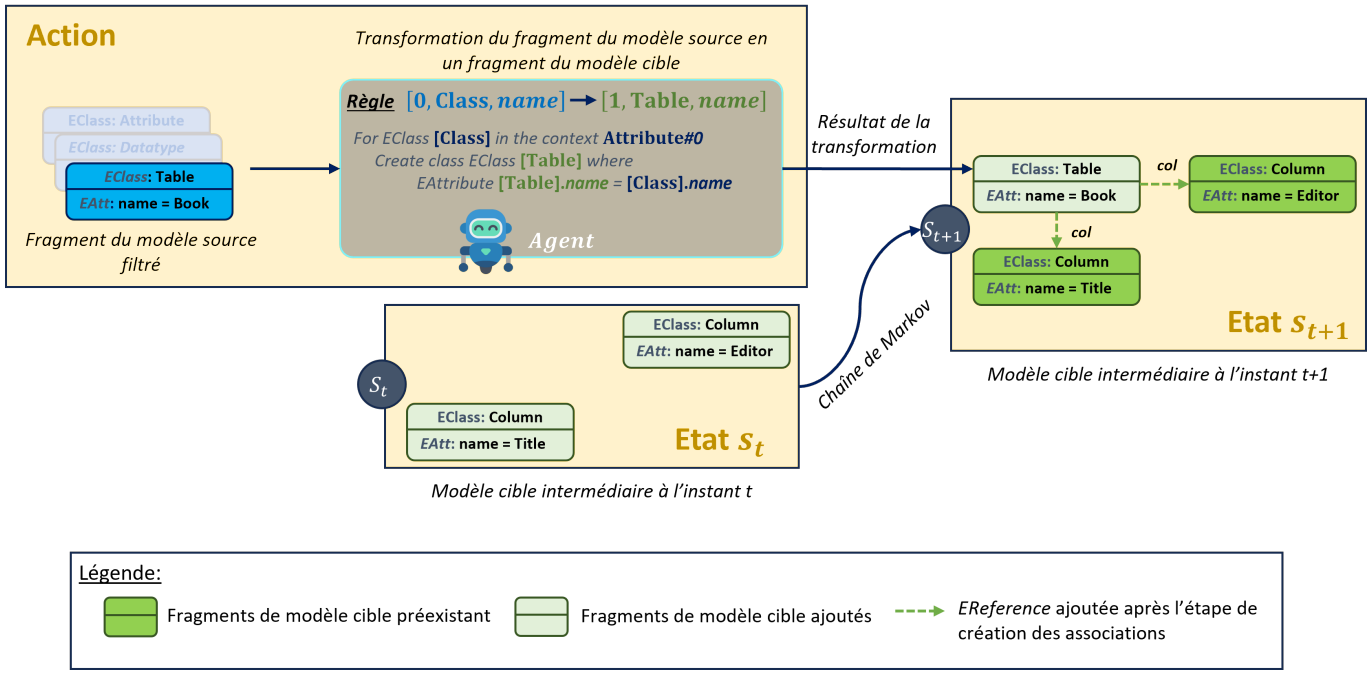


FIGURE 3.12 – Processus d'exécution d'une règle de transformation par l'agent

La chaîne de Markov représentée dans la Figure 3.12 illustre le processus d'une transition de l'agent en partant d'un état s_t (modèle cible intermédiaire à l'instant t) pour rejoindre un état s_{t+1} (modèle cible intermédiaire à l'instant $t+1$), après avoir exécuté une action (une règle de transformation). Une fois la règle de transformation exécutée par l'argent, une dernière activité consiste à utiliser le contexte #0 du pattern cible pour reconstruire les associations de type *EReference* entre les différentes *EClass* présentes dans le modèle cible intermédiaire à l'instant $t+1$. Par exemple, une fois la classe de type $[Table]$ créée, les associations de type *col* (matérialisées par les flèches vertes en pointillés) ont été ajoutées pour recréer ses relations avec les classes de type $[Column]$ préexistantes. L'agent dispose d'une mémoire dans laquelle il stocke, pour chaque épisode, la traçabilité des règles exécutées lors des actions précédentes. La mémoire de l'agent est structurée comme suit : $[P_s, T_s^a, P_c, T_c^a]$ où T_s^a correspond aux fragments de modèle source conformes au pattern source P_s qui vont être transformés

et T_c^a correspond aux fragments du modèle cible créés, conformes au pattern cible P_c. Grâce à la mémoire de l'agent concernant les précédentes règles exécutées, l'agent est en capacité de reconstruire des patterns de type $[n-m]$. Ce processus s'inspire de l'algorithme de résolution du langage ATL [88]. De ce fait, l'ordre dans lequel l'agent exécutera les règles n'aura aucune importance.

Le logigramme de la Figure 3.13 reprend l'intégralité du processus d'exécution d'une règle de transformation par l'agent.

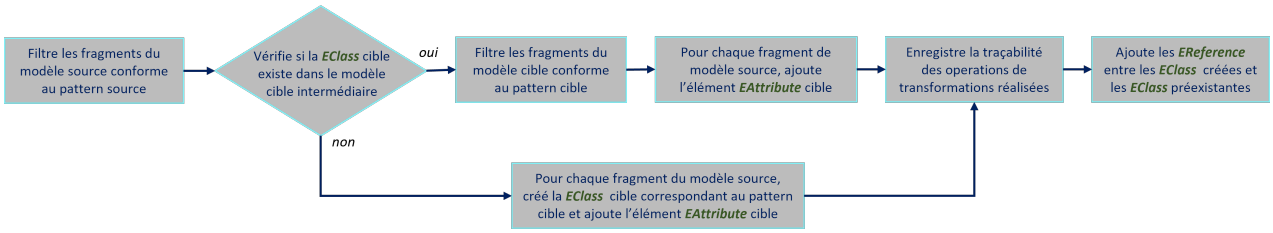


FIGURE 3.13 – Logigramme du processus d'exécution d'une règle de transformation par l'agent

3.5 Processus de transformation de modèles

Étant donné que les règles de transformation sont inférées au niveau métamodèle, la matrice de transformation peut-être directement réutilisée dans le cadre d'une transformation. Dans ce cas, pour tout nouveau modèle conforme au métamodèle source appris, il est possible de réexploiter la matrice de transformation générée par l'agent pour prédire son homologue conforme au métamodèle cible appris. La Figure 3.14 illustre le déroulement du processus de transformation de modèles.

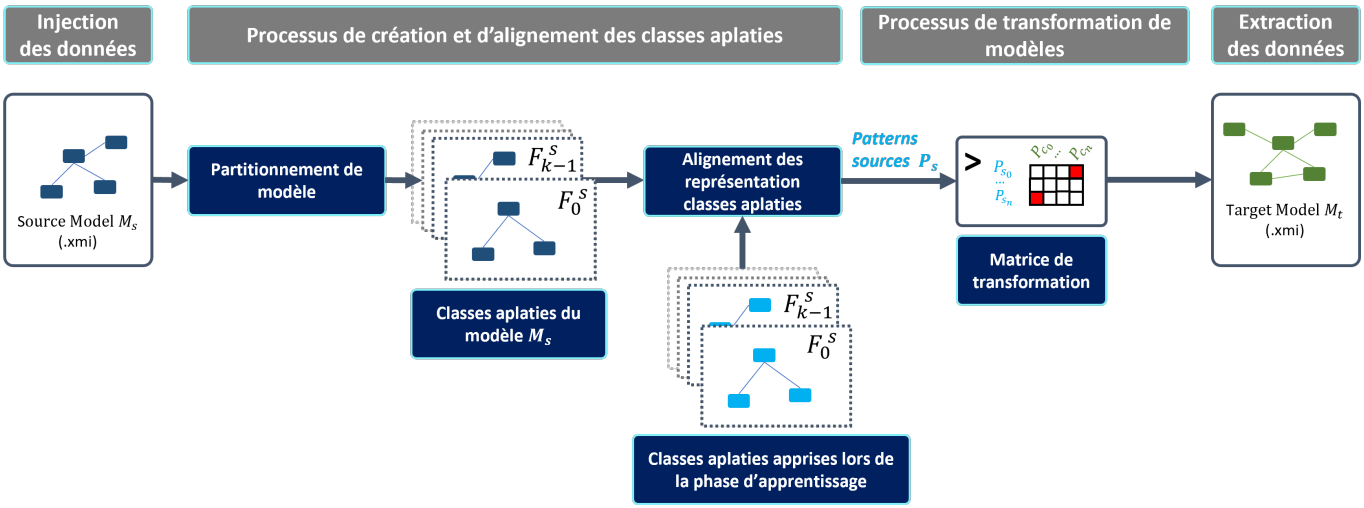


FIGURE 3.14 – Processus de transformation de modèles

Pour commencer, il est nécessaire d'identifier les patterns présents dans le nouveau modèle source M_s (au format XMI) à transformer. Pour cela, une première étape de partitionnement du modèle M_s

est réalisée pour déterminer les classes aplaties présentes dans le modèle. A l'image de la Section 3.4.2 les différents patterns sont extraits des représentations aplaties identifiées lors de la phase de partitionnement du modèle. Étant donné que la numérotation des patterns peut diffère de celle des patterns identifiés pendant la phase d'apprentissage, une étape d'alignement est nécessaire pour distinguer les patterns équivalents.

Une fois les patterns sources déterminés, le processus de transformation peut débuter. Pour chaque pattern source, l'agent choisit le pattern cible qui lui est attitré en fonction des poids de la matrice de transformation. Le processus d'exécution d'une règle de transformation suit exactement les étapes présentées en Section 3.4.3. La construction du modèle cible M_c sera terminée une fois que l'agent aura transformé tous les patterns sources présents dans le modèle source. Le fichier généré par l'agent, au format XMI, sera conforme au métamodèle cible appris pendant la phase d'apprentissage.

L'échange d'informations devient alors possible par l'intermédiaire d'un mécanisme de transformation qui traduit les messages échangés entre les systèmes. Le mécanisme de connexion présenté permet ainsi d'établir un climat propice à la communication entre les systèmes.

3.6 Synthèse

Le Chapitre 3 a permis de présenter la première contribution scientifique de ce manuscrit, à savoir, une approche d'apprentissage par renforcement des règles de transformation, qui a pour vocation d'automatiser intégralement la construction d'un connecteur pour assurer l'interopérabilité entre deux ou plusieurs systèmes. L'approche proposée s'appuie sur les principes de l'apprentissage Q, qui emploie un agent numérique pour, simultanément, aligner les concepts des métamodèles source et cible et inférer les règles de transformation qui les relient. La proposition de ce Chapitre 3 intègre à présent, la collection expérimentale proposée au chapitre précédent, et fera partie de l'étude comparative, ou *benchmark* du Chapitre 4.

La seconde contribution de ce chapitre était de proposer des grilles de caractérisation des modèles de transformation, permettant de définir les patterns de transformation qui interviennent lors d'une transformation de modèles. Les caractéristiques des modèles de transformation ont été divisées en deux catégories : les caractéristiques structurelles et terminologiques. Étant donné que les jeux de données utilisés pour valider les approches de la collection expérimentale sont tous différents, caractériser les patterns de transformation permet alors de comparer les transformations de modèles en fonction de leurs caractéristiques, et non en fonction de la sémantique de leur concepts.

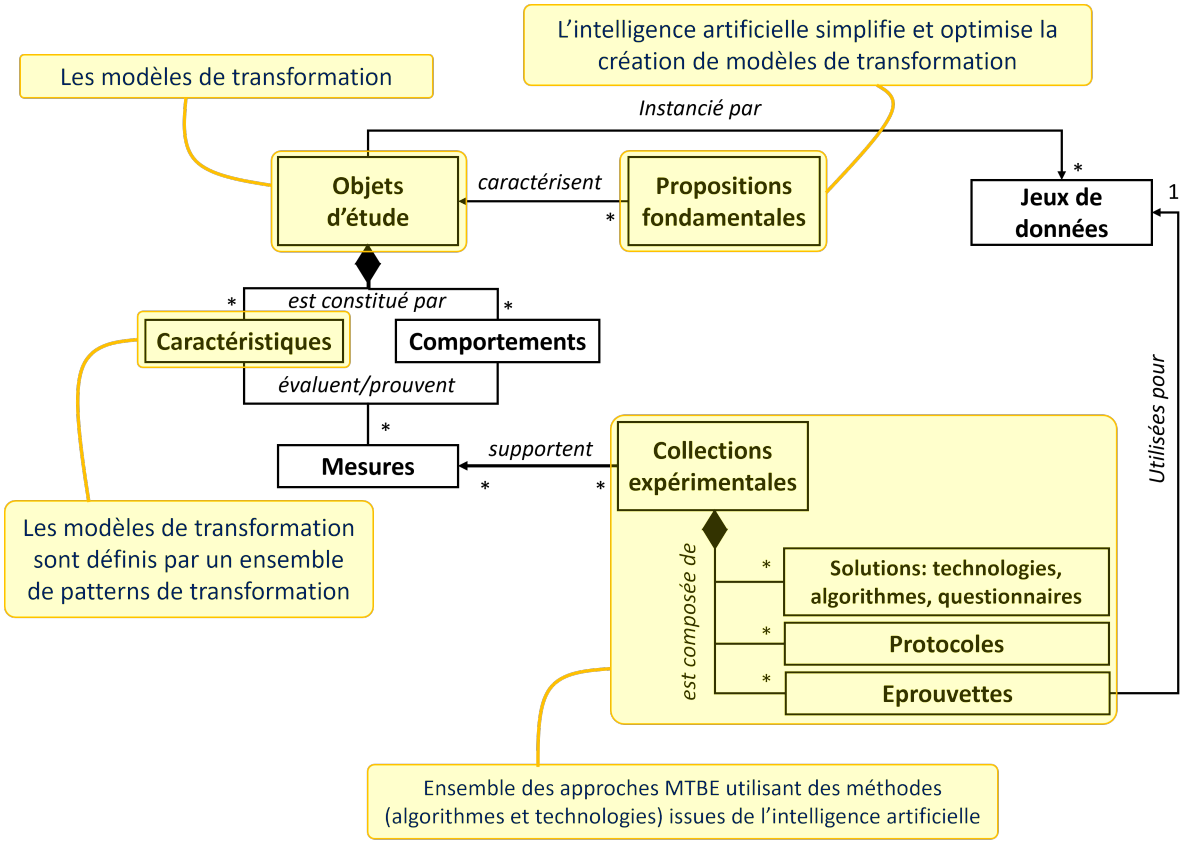


FIGURE 3.15 – Synthèse du Chapitre 3

Chapitre 4

Validation de la proposition

Contenu

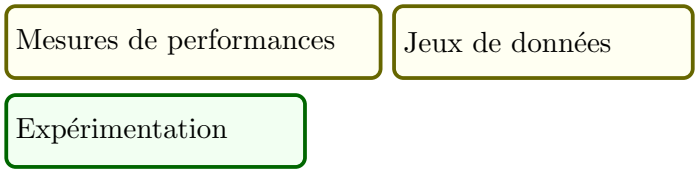
Selon la démarche d'Hevner [80], l'évaluation de la proposition fondamentale à travers un plan d'expérimentation est une activité obligatoire dans le cadre d'une conduite de recherche. Dans les chapitres précédents, une proposition fondamentale a été établie, selon laquelle, les algorithmes d'apprentissage machine permettraient de simplifier et d'optimiser la création de transformations de modèles, pour assurer une interopérabilité générique, automatique et dynamique entre les systèmes. L'objet d'étude sur lequel porte cette proposition correspond aux modèles de transformation, qui sont eux-mêmes caractérisés selon les patterns de transformation qu'ils font intervenir lors d'une transformation de modèles.

Des expérimentations seront alors menées pour valider l'approche d'apprentissage par renforcement des modèles de transformation proposée au Chapitre 3, afin d'apporter une réponse concrète aux hypothèses de la proposition fondamentale, à savoir :

(H1) Le développement d'un mécanisme de connexion automatique qui minimise les pertes sémantiques.

(H2) Le développement d'un mécanisme qui minimise les efforts humains pour créer le connecteur en termes de pré-traitement et post-traitement.

Pour cela, des mesures de performances en phase avec les hypothèses dressées en Section 1.3.3, ainsi que des jeux de données de validation doivent être identifiés.



4.1 Protocole expérimental pour la validation de la proposition

4.1.1 Matériel, logiciels et librairies utilisés

Les résultats ont été obtenus en utilisant un ordinateur portable Dell de modèle G3 3779 ayant pour configuration un processeur Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz, 2208 MHz, 6 cœurs, 12 processeurs logiques et une capacité de mémoire RAM de 16 Go.

L'algorithme et les tests de validation ont été réalisés depuis l'environnement de développement *Visual Studio Code* en python. La librairie *PyEcore*[1] (version 0.12.2) a été utilisée pour connecter l'environnement d'*Eclipse Modeling Framework* (EMF) avec l'environnement de développement *Visual Studio Code*. Le plugin EMF a permis de développer les différents métamodèles présentés (.ecore) et les différents modèles (.xmi), instanciés à partir des métamodèles précédents.

1. `https://pypi.org/project/pyecore/`

4.1.2 Paramètres de l'algorithme

Concernant les paramètres de l'algorithme, l'apprentissage Q a été exécuté sur un total de 1000 épisodes. La valeur de ϵ, permettant d'inciter l'agent à explorer son environnement ou à exploiter ses connaissances, suit une décroissance linéaire au fil des épisodes. Ainsi l'agent démarre avec un $\epsilon = 1$, signifiant que l'agent explore son environnement dès le début de l'apprentissage. Selon l'équation de Bellman, la valeur du taux d'apprentissage a été fixée à $\alpha = 0.1$. Le facteur de réduction γ, quant à lui, a été fixé à 0.9.

4.1.3 Mesures de performances

Des critères de performance ont été identifiés pour mesurer la capacité des approches de la collection expérimentale à résoudre certains types de patterns de transformation. Ces critères sont cohérents avec les hypothèses définies précédemment en Section 1.3.3. Les mesures de performances suivantes ont été utilisées pendant la phase d'apprentissage :

- Etant donné que l'objectif d'un modèle de transformation est de minimiser les pertes d'informations lors de la transformation d'un modèle, des mesures telles que le *recall*, la précision et la F-mesure ont été utilisées pour mesurer la validité des modèles obtenus après transformation. Ces indicateurs permettent d'évaluer si les règles de transformation ont correctement été apprises lors de la phase d'apprentissage ;
- Les efforts humains pour construire et valider les modèles de transformation obtenus du point de vue (1) des *pré-traitements* nécessaires, à savoir le nombre de modèles à fournir pour parfaire un entraînement optimal, ainsi que les éléments supplémentaires nécessaires tels que les pré-alignements ; (2) Des *post-traitements* obligatoires pour valider les règles de transformation obtenues ;
- La durée d'entraînement nécessaire pour apprendre les règles de transformation.

Concernant la phase de prédiction, ou de transformation, les métriques tels que le *recall*, la précision et la F-mesure seront réutilisées pour vérifier la capacité des règles de transformation précédemment inférées à transformer tout nouveau modèle conforme au métamodèle appris.

4.1.4 Jeux de données

Les transformations *Class2Relational* et *Families2Persons* présentées en Section 3.1 ont été sélectionnées pour valider l'efficacité de l'approche proposée à inférer des règles de transformation exécutables et à répondre aux enjeux de l'interopérabilité *plug and play*, à savoir la création d'un connecteur générique, automatique et dynamique. Comme expliqué précédemment, le choix de ces transformations est motivé par plusieurs raisons : (1) Les métamodèles et les règles de transformation qui les relient sont formellement définis et disponibles ; (2) Chacune de ces transformations comporte des patterns de transformation importants à étudier (voir tableau de caractérisation des jeux de données 3.6), telles que :

- La transformation *Class2Relation* nécessite une compréhension de la structure interne des

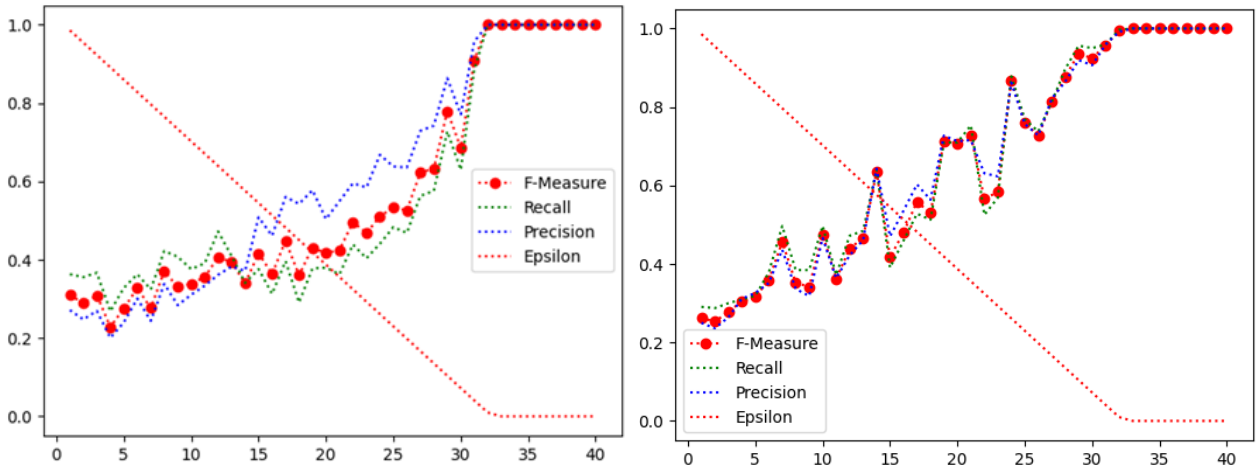
classes. La valeur d'un attribut conditionne la transformation d'une classe (σ_a). Il est donc important de distinguer les classes d'un même type en fonction des attributs qu'elles possèdent. De plus, une compréhension de la structure externe des classes est requise. Le type d'une classe adjacente à la classe étudiée conditionne sa transformation (σ_c). Il est donc important de distinguer les classes d'un même type selon leurs relations avec certaines classes adjacentes

– La transformation *Families2Persons* nécessite une compréhension des associations entre les classes. Il faut donc considérer non seulement les classes voisines, mais aussi les types d'associations pour comprendre les conditions de la transformation ((σ_r)) ;

L'objectif des expérimentations est de mesurer la capacité de l'approche proposé à résoudre les conflits structuraux et terminologiques de ces deux transformations.

4.2 Résultats

Des expérimentations ont été menées pour évaluer les performances de l'algorithme d'apprentissage Q à résoudre les conflits structuraux et terminologiques des transformations *Class2Relational* et *Families2Persons*. L'algorithme a été évalué selon les critères de performance établis en Section 4.1.3. Les résultats obtenus sont présentés dans le Tableau 4.1.



(a) Transformation *Class2Relational*

(b) Transformation*Families2Persons*

FIGURE 4.1 – Courbes d'apprentissage de l'algorithme d'apprentissage Q

Interprétation des résultats

A première vue, l'agent a bel et bien déterminé la politique optimale pour résoudre chacune des transformations proposées. Cela signifie que les règles de transformation ont parfaitement été inférées. L'apprentissage par renforcement et plus particulièrement l'algorithme d'apprentissage Q semble être une solution adaptée pour retrouver le mappage, i.e. les relations structurelles et sémantiques entre les

	Performances d'apprentissage				***Performance de transformation***
Transformation	*Temps (s)*	*P*	*R*	*F*	*F*
Families2Persons	0.76	1.0	1.0	1.0	1.0
Class2Relational	2.5	1.0	1.0	1.0	1.0

TABLE 4.1 – Résultats de l'apprentissage Q des règles de transformation

métamodèles. Cela signifie également que, pour tout type de transformations comportant les mêmes caractéristiques structurelles et terminologiques, l'algorithme d'apprentissage Q sera en mesure de retrouver l'intégralité des règles de transformation. Ces transformations seront alors dites équivalentes.

La phase de transformation, à savoir, le processus de réutilisation des règles de transformation apprises lors de la phase d'apprentissage (sous la forme d'une matrice de transformation), a été validé pour des instances de modèle conformes aux métamodèles *Class* et *Family* différentes du jeux de données utilisé pendant la phase d'apprentissage.

La Figure 4.1 montre l'évolution des indicateurs de performances, tels que la F-mesure, le *recall* et la précision, pour chacun des épisodes de la phase d'apprentissage. Les métriques de performance augmentent progressivement au fur et à mesure que l'entraînement avance et finit par atteindre la performance maximale de 1.0 (à savoir 100%) pour chacun des indicateurs. Une F-mesure égalant 100% signifie que l'ensemble des éléments attendus (classes, associations et attributs) ont été parfaitement inférés sans aucune perte (évaluée par le *recall*) et sans nuisance (évaluée par la précision). L'amélioration des performances est en partie dûe à la décroissance du coefficient Epsilon ϵ, qui signifie que l'agent ré-exploite les connaissances acquises au fil de ses expérimentations.

Pendant la phase d'entraînement, l'agent essaye différentes combinaisons de règles et tente de trouver la séquence de règles qui maximisera la F-mesure à la fin de l'épisode. L'oscillation des courbes prouve que l'agent suit une politique basée sur le principe de "teste et apprend" et expérimente différents enchaînements de règles qui se révèlent ne pas être forcément corrects. Concernant la durée de la phase d'apprentissage, les temps d'entraînements pour l'algorithme d'apprentissage Q sont tout à fait respectables pour une approche d'interopérabilité *plug and play*. Cependant, une étude plus approfondie serait à mener sur des transformations faisant intervenir un plus grand nombre d'éléments et de règles de transformation. Par ailleurs, dans cette étude, nous avons voulu montrer qu'il était possible d'utiliser l'apprentissage par renforcement pour inférer des règles de transformation complexes entre deux métamodèles, en utilisant uniquement un modèle source et un modèle cible contenant un minimum d'éléments. Pour que l'approche présentée soit efficace, il faut obligatoirement que toutes les règles de transformation soient représentées au moins une fois dans les jeux de données. En effet, l'agent ne peut apprendre les patterns de transformation qu'il ne voit pas dans son environnement. Il est alors important de se demander quel est le minimum d'éléments par modèle à fournir pour que l'apprentissage soit efficient.

Enfin, les résultats sont en partie obtenus grâce à la création de patterns sources et cibles suffisamment significatifs pour pouvoir faire correspondre la représentation aplatie de chacune des classes du modèle source avec son homologue dans le modèle cible. La représentation aplatie d'une classe semble

être une vue suffisamment détaillée pour capturer le contexte externe et interne d'une classe. Les résultats obtenus pour les transformations *Class2Relational* et *Families2Perons* sont présentés plus en détail dans les Sections 4.2.1 et 4.2.2.

4.2.1 Résultats obtenus pour la transformation *Class2Relational*

Le processus décisionnel *Markovien* de la Figure 4.2 expose le comportement de l'agent pour chaque état atteint, à savoir, pour chaque modèle cible intermédiaire généré lors de la reconstruction du modèle relationnel.

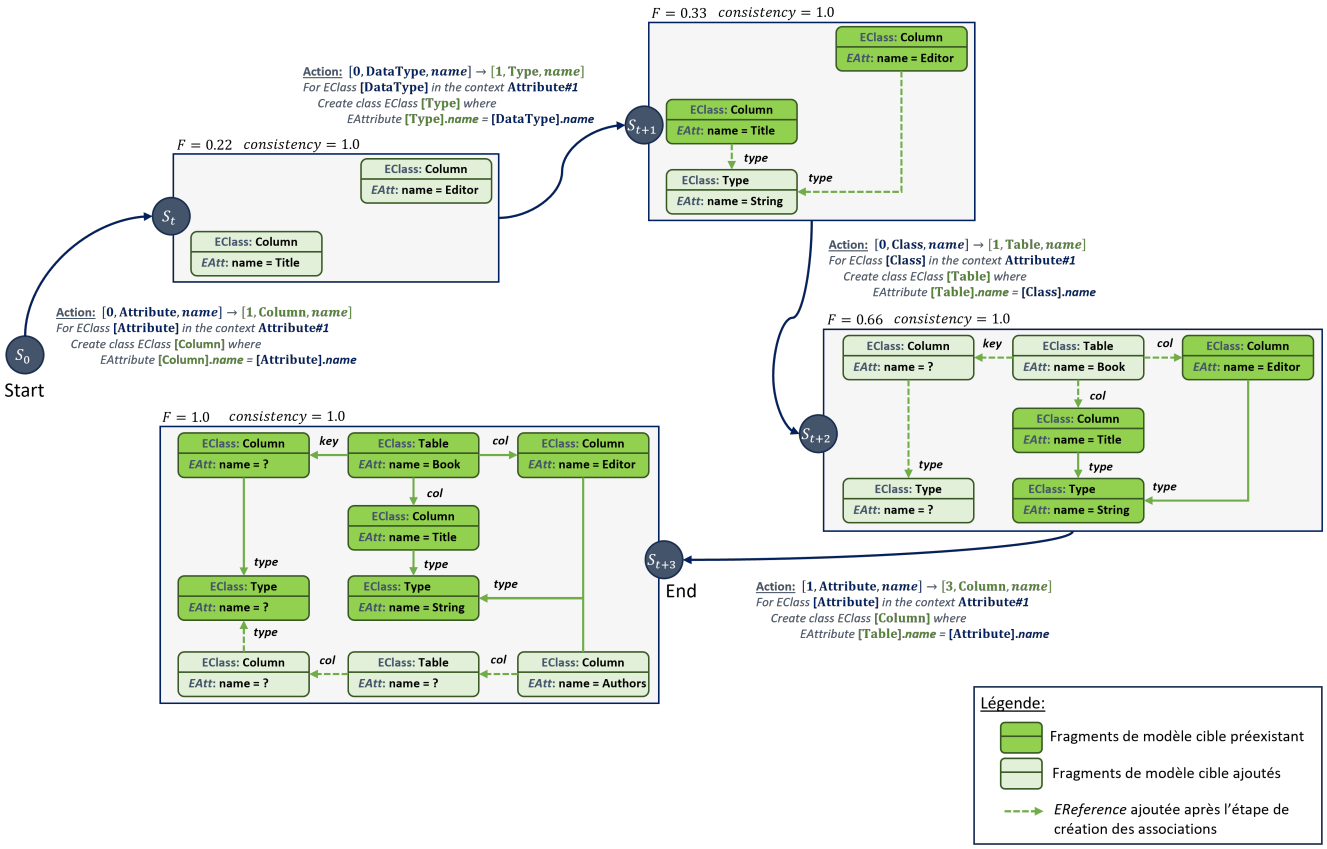


FIGURE 4.2 – Chaînes de *Markov* pour la transformation *Class2Relational*

La séquence optimale de règles de transformation, exécutées par l'agent, a permis d'édifier la matrice de transformation de la Figure 4.3, où, les lignes de la matrice correspondent aux patterns sources tandis que les colonnes correspondent aux patterns cibles. Chaque pattern source P_s est associé à un pattern cible P_c. Un couple $[P_s, P_c] = 1.0$ témoigne qu'une règle de transformation existe entre ces deux patterns. Dans le cas présent, le modèle de transformation qui permet la conversion d'un modèle de classes vers un modèle relationnel est représenté sous une forme matricielle, qui encapsule l'intégralité des règles de transformation trouvées par l'agent.

L'analyse de la matrice de transformation permet d'extraire et de valider les relations inférées entre les patterns sources et les patterns cibles. Ainsi, grâce à la matrice de transformation, il est possible de voir que la représentation aplatie Attribute#0 est liée à la représentation aplatie Column#1 par l'intermédiaire des patterns sources des lignes 1, 3 et 4 et des patterns cibles des colonnes 1, 2, 3. Chacun des couples $[P_{s_1}, P_{c_1}]$, $[P_{s_3}, P_{c_2}]$, $[P_{s_4}, P_{c_3}]$ exprime l'alignement des attributs, et par extension des classes, provenant des modèles source et cible. Il est intéressant de noter que le pattern source $[0, Attribute, multiValued]$ (ligne 2) est associé au pattern cible *None* (colonne 7), ce qui signifie que l'élément de type *EAttribute multiValued* n'existe pas dans le modèle cible.

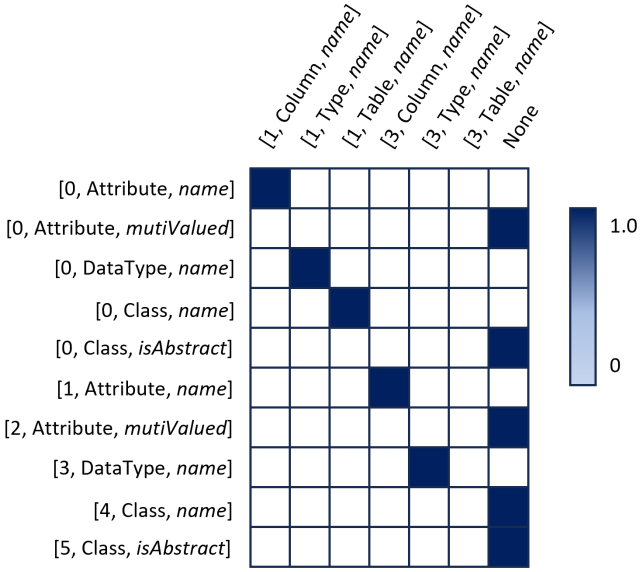


FIGURE 4.3 – Matrice de transformation pour la transformation *Class2Relational*

Analysons plus précisément la chaîne de *Markov* générée par l'agent et essayons d'expliquer l'intérêt d'utiliser la représentation aplatie des classes pour améliorer l'alignement des attributs du modèle de classes et du modèle relationnel. Pour rappel, les règles qui permettent de transformer un diagramme de classes en diagramme relationnel sont présentées dans le Tablea 3.1.

La règle (R3) concernant les *EClass* de type $[Attribute]$ ayant pour *EAttribute* $[Attribute].multiValued$ égal à *False* est représentée par le couple $[P_{s_1}, P_{c_1}]$, soit la règle de transformation $[0, Attribute, name] \rightarrow [1, Column, name]$ où, chaque *EClass* de type $[Attribute]$ appartenant au contexte Attribute#0 est transformée en *EClass* de type $[Column]$. Les triplets en attribut sources $< Attribute, name, Title >$ et $< Attribute, name, Editor >$ sont alors respectivement transformés en triplets en attribut cibles $< Colmumn, name, Title >$ et $< Colmumn, name, Editor >$. La représentation aplatie Attribute#0 permet d'aiguiller correctement la transformation des *EClass* de type $[Attribute]$. Cela qui signifie que la condition de transformation en attribut σ_a concernant les *EAttribute multiValued* a correctement été prise en compte grâce à la capture du contexte interne de la *EClass* de type $[Attribute]$ par la représentation aplatie Attribute#0.

Pour atteindre l'état s_{t+1}, l'agent exécute la règle de transformation $[0, DataType, name] \rightarrow [1, Type, name]$ correspondant à la règle (R2) où, chaque *EClass* de type $[DataType]$ appartenant au contexte Attribute#0 est transformée en *EClass* de type $[Type]$. Le triplet en attribut source $< DataType, name, String >$ est alors transformé en triplet en attribut cible $< Type, name, String >$. Une fois le fragment créé, une phase de reconstitution de la structure externe des *EClass* est réalisée. Cette étape se traduit par la création d'associations de type *EReference* entre les *EClass* préexistantes ($< Column, name, Title >$ et $< Column, name, Editor >$) et la *EClass* de type $[Type]$ qui vient d'être ajoutée au modèle cible intermédiaire. L'analyse de la représentation aplatie Column#1 permet de reconstituer le contexte des *EClass* de type $[Column]$ *Title* et *Editor* en ajoutant des associations *EReference* de type *type* avec la *EClass* de type $[Type]$ qui vient d'être créée. La création de ces associations est rendue possible par la représentation aplatie Column#1 qui exprime le contexte externe spécifique de la classe de type $[Column]$. Étant donné que les associations *EReference* ont été établies entre les bonnes *EClass*, le score de cohérence (*consistency* en anglais) est maintenu égal à 1.0.

Analysons à présent la règle de transformation $[0, Class, name] \rightarrow [1, Tabe, name]$ correspondant à la règle (R1), qui conduit à l'état s_{t+2}. La création de la *EClass* de type $[Table]$ *Book* entraîne la création d'une *EClass* de type $[Column]$ associées l'une à l'autre par l'intermédiaire de la *EReference* de type *key*. La valeur de l'*EAttribute* intégré à l'*EClass* de type $[Column]$ sera initialisée à " ?" car il s'agit d'une valeur inconnue par le modèle source. Étant donné que la *EClass* de type $[Table]$ est considérée comme une caractéristique héritée de la représentation aplatie Column#1, et qu'elle dispose de sa propre représentation aplatie Table#4 (voir représentation aplatie du modèle relationnel de la Figure 3.11), alors la *EClass* de type $[Table]$ est en charge de la reconstruction de son propre contexte, à savoir, de la construction de sa structure interne et externe. Ainsi, l'analyse de la représentation aplatie Table#4 permet la construction de la *EClass* de type $[Column]$ " ?". De la même façon, la *EClass* de type $[Column]$ dispose de sa propre représentation aplatie Column#0. La création d'une *EClass* de type $[Column]$ " ?" entraîne la création d'une *EClass* de type $[Type]$ " ?" où la *EReference* de type *type* associe les deux *EClass*. Une fois de plus, seul l'analyse de la représentation aplatie Column#0 a permis la construction de la *EClass* de type $[Type]$.

La règle (R1) est le parfait exemple pour démontrer l'implication des représentations aplaties des classes dans le processus de création du modèle cible. La construction du modèle cible suit donc un processus de création de proche en proche guidé par les représentations aplaties.

Pour résumer, l'agent tente d'aligner les attributs des métamodèles source et cible en fonction du contexte dans lequel ils interviennent. Le contexte d'un attribut est exprimé par les représentations aplaties de la classe auquel il appartient. De cette manière, l'alignement d'un attribut source et d'un attribut cible entraîne également l'alignement de la classe source et de la classe cible qui contiennent les deux attributs. Une fois les attributs alignés, les classes aplaties sont exploitées pour recréer le contexte de l'attribut, à savoir la création d'attribut spécifique à la classe, ou encore, la création de références et d'associations entre les classes.

4.2.2 Résultats obtenus pour la transformation *Families2Persons*

Concernant la transformation *Families2Persons*, la matrice de transformation éditée par l'argent est présentée dans la Figure 4.4. Son analyse permet de constater que les représentations aplaties Member#0 et Member#2 sont reliées à la représentation aplatie Woman#1 par l'intermédiaire des couples patterns source-cible $[P_{s_1}, P_{c_1}]$, $[P_{s_2}, P_{c_2}]$ et $[P_{s_5}, P_{c_1}]$, $[P_{s_6}, P_{c_2}]$. La relation entre la représentation aplatie Member#0 et la représentation aplatie Woman#1 exprime la règle de transformation (R2), tandis que la relation entre la représentation aplatie Member#2 et la représentation aplatie Woman#1 exprime la règle de transformation (R4).

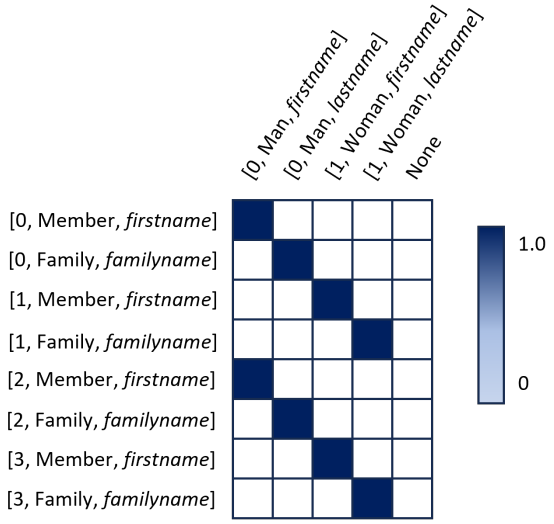


FIGURE 4.4 – Matrice de transformation pour la transformation *Families2Persons*

Ainsi les représentations aplaties Member#0 et Member#2 permettent de capturer la condition de transformation placée sur la relation σ_r entre les *EClass* de type $[Family]$ et $[Member]$, et, *in fine*, de guider correctement la transformation des patterns sources vers les bons patterns cibles. De cette façon, les couples patterns source-cible permettent d'aligner l'*EAttribute firstname* de la *EClass* de type $[Member]$ du modèle source avec l'*EAttribute firstname* de la *EClass* de type $[Woman]$ du modèle cible, ainsi que l'*EAttribute familyname* de la *EClass* de type $[Family]$ du modèle source avec l'*EAttribute lastname* de la *EClass* de type $[Woman]$ du modèle cible. L'alignement des attributs est rendu possible grâce la première dimension de chaque pattern, qui spécifie le contexte dans lequel intervient l'attribut, et par extension, la classe qui le contient.

Le processus décisionnel *Markovien* de la Figure 4.5 expose le comportement de l'agent pour chaque état atteint, à savoir, pour chaque modèle cible intermédiaire généré lors de la transformation *Families2Persons*. L'analyse des chaînes de *Markov* permet une fois de plus de cerner l'importance de la représentation aplatie des classes. Lorsque l'agent applique successivement les règles de transformation $[2, Member, firstname] \rightarrow [0, Man, firstname]$ et $[2, Family, familyname] \rightarrow [0, Man, lastname]$ générant respectivement les fragments $< Man, firstname, Tyrion >$, $< Man, firstname, Jaime >$

puis $< Man, lastname, Lannister >$, $< Man, lastname, Lannister >$, nous pouvons constater que seul deux *EClass* de type $[Man]$ sont créées au lieu de quatre. L'agent s'appuie à la fois sur les représentations Member#2 et Man#0 pour reconstituer le contexte interne de la *EClass* de type $[Man]$ en intégrant convenablement les couples d'*EAttribute* *firstname* et *lastname* au sein de la même *EClass*.

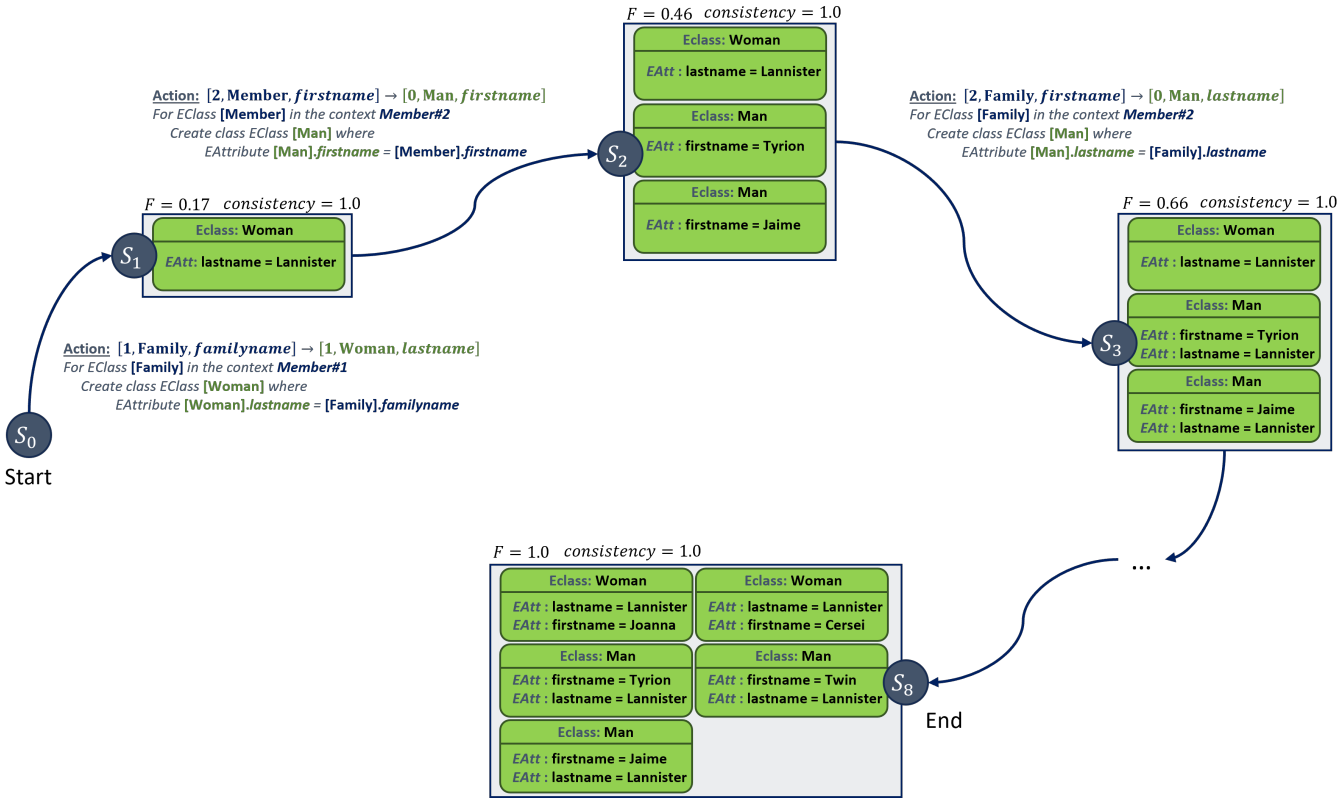


FIGURE 4.5 – Chaînes de *Markov* pour la transformation *Families2Persons*

4.3 *Benchmark*

Une étude comparative a été menée afin de mesurer les performances de chacune des approches MTBE de la collection expérimentale (voir Tableau 2.3). La synthèse des résultats suit 4 niveaux d'analyse :

Niveau 0 : Les résultats sont difficilement exploitables ou ne sont pas clairement définis ;
Niveau 1 : Les résultats sont exploitables mais des calculs ont été réalisés ;
Niveau 2 : Les résultats ont été repris tel qu'ils sont présentés dans l'article source ;
Niveau 3 : Chacun des algorithmes a été exécuté pour obtenir les résultats.

Le *benchmark* proposé a pour objectif d'évaluer puis de comparer les performances des différentes approches MTBE, permettant de dériver des règles de transformation fonctionnelles. Cette étude comparative a pour objectif de mesurer l'amélioration de la qualité des approches d'apprentissage des

règles de transformation, et d'identifier les lacunes et les axes d'optimisation à suivre dans de futurs travaux. Ce *benchmark* a aussi vocation à aider de potentiels utilisateurs à prendre des décisions éclairées lorsqu'ils choisissent entre différentes solutions.

Le Tableau 4.2 synthétise la comparaison entre les différents résultats récupérés dans la littérature et l'approche proposée dans ce manuscrit. Les jeux de données utilisés pour comparer les approches sont les transformations *Class2Relational* et *UML2Relational* [46]. Comme ces deux ensembles de données ont les mêmes caractéristiques structurelles et terminologiques (à l'exception du pattern c2), la comparaison des approches MTBE de la collection expérimentale peut être réalisée (voir Tableau 3.6 pour la caractérisation des jeux de données). D'après le Tableau 4.2, concernant les acronymes de la colonne consacrée aux données d'entrée des approches, M_s correspond au modèle source, M_c correspond au modèle cible, P correspond aux pré-alignement. Pour chaque indicateur, un symbole "?" signifie que la valeur n'a pas pu être déterminée.

Approches	***Entrées***	***Performances d'apprentissage***				***Transformation***	***niveau***
		Temps (s)	*P*	*R*	*F*	*F*	
ATL [88]	$M_s + M_t$	?	1.0	1.0	1.0	1.0	3
ILP [14]	$M_s + M_t + P$	?	?	?	?	?	0
RCA [145]	$M_s + M_t + P$	?	0.9	0.8	0.85	0.93	1
GA [60]	$M_s + M_t$	?	0.75	?	?	?	2
GA [61]	$M_s + M_t$	3600	?	1.0	?	1.0	2
PSO+SA [93]	$M_s + M_t + P$	20 to 50	?	0.93	?	?	2
GA [13]	$M_s + M_t + P$	?	0.94	0.94	0.94	0.94	1
LSTM [33]	$750M_s$ $750M_t$	+4000	1.0	1.0	1.0	1.0	3
Q-learning	$1M_s$ $1M_t$	**2.5**	**1.0**	**1.0**	**1.0**	**1.0**	3

TABLE 4.2 – Benchmark des approches MTBE de la collection expérimentale

L'objectif de cette étude est non seulement de comparer les approches permettant l'apprentissage des règles de transformation grâce aux technologies de l'intelligence artificielle, mais aussi de comparer les performances de ces approches vis-à-vis des méthodes descriptives. Pour cela, les règles de transformation pour le cas d'étude *Class2Relational* ont été éditées par l'intermédiaire du langage de transformation descriptif ATL.

4.4 Discussion

Les approches sont évaluées sous deux angles. Le premier est une étude quantitative qui tentera de répondre à l'hypothèse (H1). Le second est une étude qualitative qui répondra à l'hypothèse (H2). L'objectif de cette discussion est d'apporter des éléments de réponses quant au choix de l'une des solutions évaluées dans le cadre du *benchmark*, mais aussi d'identifier les points d'amélioration qui devront être fixés pour répondre pleinement aux enjeux de l'interopérabilité des systèmes de l'Industrie

4.0.

Concernant l'hypothèse (H1) qui concerne la **construction d'un mécanisme de connexion automatique qui minimise les pertes sémantiques lors de la traduction des messages échangés**, les approches basées sur des réseaux de neurones LSTM [33], sur l'apprentissage par renforcement [29] et sur les principes de la programmation génétique [61] atteignent un score de *recall* égal à 100%. Cela signifie que les règles de transformation sont intégralement apprises et que les pertes sémantiques lors de la transformation des modèles sont nulles. Cependant, aucune indication sur le score de F-mesure n'est donnée pour les résultats obtenus par Faunes *et al.* [61]. Par conséquent, nous ne pouvons pas déterminer si les règles dérivées contiennent des dysfonctionnements qui réduisent le score de précision. L'utilisation du langage de transformation ATL atteint également un score de F-mesure égal à 100%, cependant, la génération des règles de transformation n'est pas automatique, propriété non négligeable d'une interopérabilité *plug and play*.

Par rapport à l'hypothèse (H2) qui concerne les **efforts humains nécessaires en terme de pré et de post traitement**, les approches qui atteignent un score de F-mesure égal à 100% ne nécessitent pas de post-traitement. En revanche, les approches avec une précision différente de 100% créent des perturbations pendant la phase de transformation, ce qui signifie que les modèles de sortie doivent être affinés pour supprimer les incohérences. Avec un score de *recall* inférieur à 100%, cela signifie que le modèle de sortie est incomplet après la transformation et doit être, par conséquent, complété avec les règles manquantes engendrant inévitablement des post-traitements. Concernant les efforts de pré-traitement, les approches [14, 145, 93, 13] exploitent des pré-alignements pour guider l'apprentissage des règles de transformation. Or ces pré-alignements ne sont pas toujours évidents à déterminer et constituent une source d'informations supplémentaires à fournir à l'algorithme qui n'est pas toujours disponible. De plus, l'approche LSTM de Burgueño *et al.* [33] nécessite 750 modèles sources et 750 modèles cibles pour entraîner l'architecture de traduction encodeur/décodeur. Il n'est pas toujours facile d'obtenir autant de données d'entraînement, surtout dans un contexte industriel où les données sont souvent volatiles. Concernant l'approche ATL [88], les règles de transformation sont éditées à la main par un développeur qui est sensé connaître la sémantique des métamodèles source et cible à relier. Le processus d'édition des règles de transformation est un processus long qui est souvent source d'erreurs. Par ailleurs, maintenir des règles de transformation descriptives en état de fonctionner dans un environnement hautement dynamique est une activité complexe.

Enfin, les approches qui ne génèrent pas de règles de transformation sont peu malléables. Dans le cas des réseaux LSTM et de l'apprentissage Q [29], cela signifie qu'il est difficile d'améliorer ou d'optimiser les règles de transformation apprises lorsque le score de F-mesure est différent de 100%. Dans le cas de l'apprentissage Q, selon les transformations de modèles apprises, les dimensions de la matrice de transformation peuvent inévitablement croître et complexifier l'analyse des règles de transformation obtenues. De la même façon, les réseaux de neurones LSTM sont des boîtes noires difficiles à analyser et à comprendre.

Concernant les trois grands enjeux de l'interopérabilité des systèmes dans un contexte d'Industrie 4.0 :

– La **propriété de généricité** du connecteur est garantie par l'ensemble des approches. Les ap-

proches de la collection expérimentale sont utilisables pour n'importe quel type de transformation de modèles, et ce, quelque soit la structure et la sémantique des métamodèles. Cependant, les approches qui génèrent des règles de transformation contraignent le développeur à utiliser un langage de transformation particulier comme ATL ou JESS.

- La **propriété d'automatisation** de la création d'un connecteur sur mesure, i.e. propre à la communication entre deux ou plusieurs systèmes est garantie pour les solutions ayant un score de F-mesure égal à 100%. Il est important de garder à l'esprit que chacune des solutions de l'approche expérimentale est en capacité de résoudre uniquement les conflits structuraux et terminologiques pour lesquelles elles ont été conçues. Par conséquent, des efforts de pré et de post-traitements peuvent être nécessaires, réduisant l'automatisation de la création du connecteur.
- La **propriété de dynamicité** du connecteur n'a été adressée par aucune des solutions de la collection expérimentale. Dans le cas où un connecteur doit absorber les changements et les évolutions de l'un des systèmes, les solutions actuelles proposent de réexécuter la phase d'apprentissage pour intégrer les nouvelles règles de transformation. En soi, ce n'est pas une limitation pour les approches où les délais d'apprentissages des règles de transformation sont faibles. En revanche, c'est une limitation avérée pour les approches utilisant de nombreux modèles sources et cibles, ainsi que des pré-alignements pour entraîner à nouveau le mécanisme d'apprentissage à prédire de nouvelles règles de transformation.

Enfin, pour conclure sur l'intérêt des approches d'apprentissage des règles de transformation vis-à-vis de l'approche descriptive ATL, plusieurs questions méritent d'être soulevées :

(1) Pour commencer :*les approches d'apprentissage des règles transformation par les technologies de l'intelligence artificielle sont elles plus efficaces ?*

(2) La première question entraîne inévitablement la seconde question : *les approches mixtes, hybrides, employant à la fois une expertise humaine et une aide à la décision provenant de l'intelligence artificielle ne seraient-elles pas un bon compromis ?*

Pour répondre à la question (1), les approches par apprentissage, et plus particulièrement les approches exploitant des réseaux de neurones [33], des algorithmes génétiques [61], ou encore des algorithmes d'apprentissage par renforcement comme présentés dans ce manuscrit, semblent être des solutions efficaces pour inférer des patterns de transformation faisant intervenir des conditions complexes que l'humain aurait du mal à appréhender. En plus d'apporter probablement un gain de temps vis-à-vis du temps nécessaire pour éditer à la main un modèle de transformation en langage ATL, les approches d'apprentissage sont automatiques. Cependant, les approches par apprentissage sont intéressantes, pourvu que la quantité d'éléments à fournir à l'algorithme d'apprentissage ne soit pas trop importante. Un compromis est donc nécessaire entre la mise en oeuvre de l'algorithme d'apprentissage machine et la performance finale de l'apprentissage.

Enfin, concernant la question (2), Il semblerait que les algorithmes d'apprentissage atteignent leurs limites dans les situations où les êtres humains ont un avantage. En d'autres termes, il est important de reconnaître les forces et les limites de chaque approche (humaine et algorithmique) et de les utiliser de manière complémentaire. Si un algorithme fait 90% du trajet de manière automatique, il ne reste alors plus que 10% du trajet à accomplir par l'humain. Les approches hybrides alliant la puissance des

algorithmes d'apprentissage automatique au savoir faire et à l'expertise d'un humain pourraient être des solutions intéressantes à exploiter pour résoudre les problèmes qui consistes à éditer des modèles de transformation complexes.

4.5 Limitations et axes d'amélioration de la proposition

L'apprentissage par renforcement, et plus précisément l'apprentissage Q, semble être une solution très intéressante pour aboutir à la création de mécanismes de connexion génériques, automatiques et dynamiques en phase avec le contexte d'Industrie 4.0. Cependant différentes limitations méritent d'être levées dans de prochaines expérimentations.

Représentation des classes. Pour commencer, la performance de l'apprentissage par renforcement repose essentiellement sur la capacité à décrire les caractéristiques qui représentent l'environnement [121]. Dans notre étude, nous avons représenté chaque classe par une forme aplatie qui capture sa structure interne et externe. Les représentations aplaties générées sont ensuite utilisées pour construire des patterns sources et cibles qui permettront d'éditer une collection de règles de transformation testable par l'agent pendant le processus d'apprentissage. Or l'édition des patterns dépend principalement de la représentation aplatie des classes qui spécifie le contexte dans lequel intervient un attribut et sa classe. Une représentation aplatie semble beaucoup trop simpliste lorsqu'il s'agit de tenir compte des dépendances plus lointaines entre les classes (supérieures à l'ordre 2). Par ailleurs, la fonction d'extraction de la représentation des classes est un processus heuristique spécifié par un développeur. De ce fait, de nombreuses caractéristiques essentielles ne sont pas capturées lorsqu'il s'agit d'établir une représentation fidèle des classes (par exemple l'orientation des relations entre les classes...). Par conséquent, une représentation aplatie trop peu détaillée pourrait engendrer la création de patterns trop peu significatifs, et donc, contribuerait à l'édition de règles de transformation incohérentes, mauvaises où incomplètes

Avec l'essor fulgurant qu'a connu le domaine de l'apprentissage de représentation des graphes de données grâce aux réseaux de neurones [74], nous avons l'intuition qu'exploiter ces nouvelles techniques, pour obtenir une représentation des classes plus étendue et plus précise, semble être une direction de recherche à privilégier. Les GNNs ont déjà apporté des solutions concrètes aux problèmes d'interopérabilité des systèmes, notamment lorsqu'il s'agit de capturer des relations d'équivalence entre deux bases de données issues de différents silos. Le cloisonnement des données entrave la collaboration et le partage d'informations au sein d'une organisation. Koutras *et al.* [97], proposent alors une méthode pour fédérer différents silos de données, qui s'appuie sur un modèle de prédiction efficace basé sur le pouvoir de représentation des GNNs. Un GNN est ainsi utilisé pour prédire des relations d'équivalence entre les colonnes des bases de données présentes dans différents silos. Les travaux de Hao *et al.* [76] exploitent les GNNs pour leur capacité à capturer la structure hiérarchique des concepts des ontologies afin de favoriser leur alignement. Les ontologies présentent souvent des structures hiérarchiques descendantes, qui organisent systématiquement les concepts en catégories et sous-catégories. Il est donc important d'intégrer à la représentation d'un concept, la structure hiérarchique dont il dépend.

Validation et évolution de la transformation L'approche proposée génère en sortie une matrice de

transformation qui peut s'avérer être difficile à valider dans le cas où le nombre de couples pattern source-cible serait élevé. Aucune règle de transformation n'est formellement inférée (en pseudo code, ou dans un langage de transformation particulier). Cela implique que, si les modèles source et cible venaient à évoluer ou être modifiés, incorporer de tels changement dans une matrice de transformation serait une activité compliquée, si ce n'est impossible à réaliser.

L'approche proposée doit répondre à la propriété de généricité de l'interopérabilité des systèmes dans un contexte d'Industrie 4.0, à savoir, le connecteur ne doit pas être dépendant des systèmes qu'il relie, et ne dépend pas non plus d'un quelconque langage de transformation. A l'image du métamodèle d'alignement proposé par [113], une représentation structurée des règles de transformation inférées pourrait simplifier à la fois la validation et la correction des règles de transformation, ainsi que l'évolutivité du connecteur en cas d'ajout, de suppression ou de modification de nouvelles règles.

Par ailleurs les modèles de données, tout comme les graphes de données, sont des représentations éphémères des données. Ainsi les classes d'un modèle, tout comme les entités d'un graphe, sont sujettes à des évolutions. Par conséquent, la représentation vectorielle des entités, précédemment apprise, est également sujette à évoluer. C'est en ce sens que de récentes études ont pour objectif d'encoder les évolutions temporelles des graphes de données, à savoir, les modifications apportées aux entités au court du temps (ajout d'attributs, changements de valeur, ou encore ajout d'une entité voisine...). Pareja *et al.* utilisent alors un réseau de neurones récurrents (*Recurrent Neural Network* RNN) pour mettre à jour les paramètres du réseau GCN précédemment appris, afin de capturer le dynamisme des graphes. De la même façon, étant donné que la représentation vectorielle des entités évolue, alors le mappage qui relie deux graphes de données à tendance lui aussi à évoluer. Yan *et al.* [188] introduisent alors un nouveau type de problème qui consiste à aligner des graphes de connaissances dynamiques, dont le principal défi est de savoir comment mettre à jour efficacement la représentation vectorielle des entités pour des topologies de graphes évolutives.

Extensibilité de la transformation. Un nombre élevé de patterns sources et de patterns cibles pourrait être un facteur limitant pour plusieurs raisons : (1) Le temps de calcul nécessaire pour naviguer à travers la Q-table lors des phases d'apprentissage et de transformation ; (2) Une absence de généralisation des règles de transformation inférées. Une architecture extensible et évolutive est alors nécessaire pour traiter des métamodèles comportant un grand nombre de classes, et pour absorber les changements au sein des modèles.

Une discipline naissante s'intéresse aux problèmes de transformation, ou de traduction des graphes. Largement inspiré du domaine de la *traduction automatique neuronale du langage*, exploitant une architecture encodeur/décodeur pour traduire une phrase dans un langage source vers un langage cible [11], la *transformation de graphe* permet d'apprendre le mappage de transformation entre deux graphes issus de domaines différents [69], en exploitant les méthodes d'apprentissage de représentation des graphes. Des approches très prometteuses ont associé la puissance des algorithmes d'apprentissage de représentation des graphes, avec les principes de l'apprentissage par renforcement pour la génération de graphes de données, tout en limitant le corpus de modèles donné en entraînement [189].

Dans de futurs travaux, une importance toute particulière sera donnée aux approches permettant la transformation/génération de graphes de données, afin de déterminer dans quelle mesure ces mé-

thodes peuvent résoudre les limitations identifiées précédemment et répondre aux grands enjeux de l'interopérabilité des systèmes.

4.6 Synthèse

Pour finaliser la démarche de recherche de ce manuscrit, un protocole expérimental est proposé à la communauté scientifique. Ce protocole contient des jeux de données caractérisés, ainsi que des mesures de performance permettant de valider l'habilité des approches de la collection expérimentale à résoudre certains patterns de transformation identifiés (voir Figure 4.6).

Les résultats obtenus pour les jeux de données *Class2Relational* et *Families2Persons* ont été détaillés, et leur étude a prouvé que l'apprentissage par renforcement des règles de transformation est une approche fonctionnelle pour inférer les règles de transformation adéquates pour ces deux jeux de données.

Suivant le protocole expérimental, une étude de type *benchmark* a ensuite été réalisée pour évaluer puis comparer les performances des différentes approches vis à vis des hypothèses dressées au Chapitre 3. Le *benchmark* a révélé que l'approche d'apprentissage par renforcement proposée dans ce manuscrit répond en partie aux exigences d'une interopérabilité générique, automatique et dynamique. Cependant des améliorations concernant de la représentation aplatie des classe, de la validation et de l'évolutivité des règles de transformation inférée, ainsi que de l'extensibilité de l'architecture, doivent être réalisées. Les limitations identifiées nous ont conduits à déterminer l'orientation de nos recherches du côté des algorithmes d'apprentissage de représentation des graphes et des algorithmes de génération et de transformation de graphes.

A présent que l'approche d'apprentissage Q des règles de transformation a prouvée son efficacité sur des jeux de données de validation de "laboratoire", il est maintenant temps de procéder à un véritable passage à l'échelle, à savoir, la résolution de cas d'étude industriel concret.

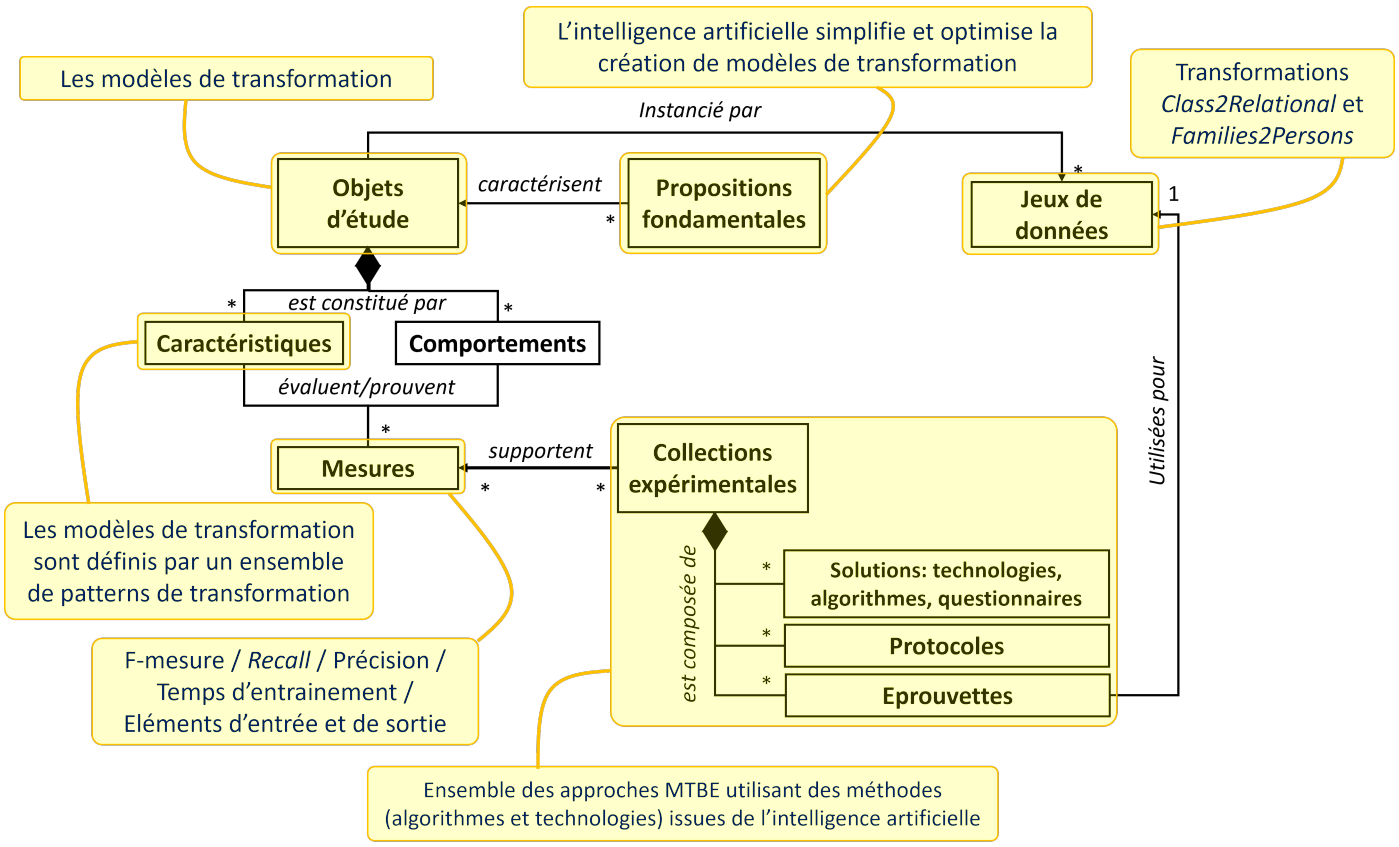


FIGURE 4.6 – Synthèse du Chapitre 4

Chapitre 5

Passage à l'échelle : transfert technologique vers l'industrie

Contenu

Le passage à l'échelle, où processus de transition d'une solution validée en milieu académique vers une mise en oeuvre à plus large échelle dans un contexte concret, est une étape importante pour confirmer que la solution proposée dans ce manuscrit, répond clairement aux besoins initialement formulés par l'environnement.

Dans le cas de cette étude, rappelons que l'environnement est défini par les concepts et les technologies de l'Industrie 4.0, tandis que ses besoins correspondent à la nécessité de construire des mécanismes de connexion génériques, automatiques et dynamiques, pour garantir l'intéropérabilité entre les systèmes qui composent une entreprise. Le contexte concret, quant à lui, fait référence à une portion de l'environnement industriel dans laquelle devra s'imbriquer la solution proposée dans ce manuscrit, à savoir, un mécanisme de connexion basé sur l'apprentissage par renforcement des modèles de transformation. Deux contextes concrets, ou cas d'étude, issus de l'industrie du spatiale, ont été identifiés pour démontrer la faisabilité de la solution proposée : (1) L'automatisation de la construction des documents de type *As-Built* ; (2) La planification automatique des activités d'Assemblages, d'Intégrations, et de Tests des satellites. Tous deux comportent des systèmes de la production de type *Manufacturing Execution System* (MES), *Enterprise Resources Planning* (ERP) et autres progiciels et supports d'informations dont la communication est requise pour mener à bien leurs activités.

Avant tout, il est important de se demander comment le mécanisme de connexion développé pourrait s'intégrer dans un environnement numérique industriel pré-existant. Quelles sont les technologies actuelles qui permettent d'aboutir à l'architecture *plug and play* tant souhaitée ? Sans oublier, comment combiner ces technologies avec une architecture MDA ? C'est en ce sens que le Chapitre 5 débutera par une présentation de l'architecture logicielle *plug and play* utilisée, ainsi que des composants développés propres à l'architecture MDA. La présentation de l'architecture logicielle utilisée sera évidemment centrée sur l'intégration du mécanisme de connexion présenté au Chapitre 3, à savoir, un modèle de transformation qui assure la fonction d'interopérabilité sémantique entre les systèmes en traduisant les messages qu'ils échangent.

Le transfert de technologies issues de la recherche, vers l'industrie, est un maillon indispensable dans le déploiement de l'Industrie 4.0. Ces technologies ont le potentiel de créer de la valeur, d'améliorer la compétitivité des entreprises, et surtout, de guider les entreprises vers une politique en phase avec les grands enjeux environnementaux et sociaux d'aujourd'hui.

5.1 Architecture logicielle *plug and play*

L'architecture logicielle joue un rôle crucial dans la capacité d'un système de systèmes à évoluer de manière durable à travers le temps. Une bonne conception architecturale facilite à la fois son extensibilité en favorisant l'intégration de nouvelles technologies et de nouvelles sources d'informations, son adaptabilité aux changements et aux évolutions, ainsi que sa modularité de façon à accroître le couplage entre les systèmes.

Les systèmes de systèmes, ou application (*Apps*) dans le cas présent, sont conçus pour réaliser une succession de tâches, de sorte à accomplir des objectifs spécifiques. Une application s'appuie sur différents composants logiciels, systèmes ou services, pour aboutir à la tâche qui lui est attitrée.

Historiquement, les architectures logicielles sont conçues sous forme de monolithe, i.e. comme une seule entité indépendante qui regroupe l'ensemble des fonctionnalités, des composants logiciels et des modules au sein d'un unique code source. Cependant, les architectures monolithiques limitent l'extensibilité et la reconfigurabilité des applications. A mesure que la complexité des processus, des métiers et des produits évoluent, la complexité des systèmes numériques croît également et devient de plus en plus difficile à maintenir et à faire évoluer. Lorsque l'interopérabilité des systèmes qui composent une application se trouve menacée, c'est toute la coopération au sein de l'application qui se trouve compromise.

Pour réaliser la vision d'une industrie hyperconnectée, flexible, évolutive et réactive aux changements, l'architecture logicielle de l'Industrie 4.0 doit être en mesure de s'adapter automatiquement, d'absorber les changements sans compromettre le fonctionnement général de l'architecture, et ce, avec un minimum d'effort humain. Les architectures orientées agents et services apportent des concepts et des technologies concrètes pour résoudre les problèmes de modularité, d'intéropérabilité et de coopération entre les systèmes au sein d'une application. Elles permettent, à ce jour, l'édification de systèmes complexes robustes et durables dans le temps, qui répondent aux propriétés de l'Industrie 4.0 [79] et favorisent l'émergence de systèmes CPS.

Les architectures multi-agents. Les architectures orientées agents permettent le développement de systèmes industriels distribués et intelligents, qui facilitent les opérations de planification des ressources de fabrication, d'ordonnancement et de contrôle de l'exécution [149]. Un agent encapsule les systèmes logiciels existants au sein de l'entreprise, les ressources manufacturières tels que les travailleurs, les machines, les outils, ainsi que les produits, les pièces et les opérations. Dans une architecture multi-agents, chaque agent autonome est assigné à un activité. La propriété de décentralisation permet à l'agent de décider par lui-même des actions, à mener en fonction des données acquises grâce à ses interactions avec d'autres agents au sein d'un réseau connecté [54].

Les architectures orientées services (*Service Oriented Architecture* SOA). Un SOA est une architecture modulaire qui encourage la réutilisation de composants logiciels, ou services, autonomes et indépendants, accessibles à partir de leur interface [55]. Les SOA permettent de maîtriser l'extensibilité et la complexité croissante des systèmes, en décomposant leurs modules en un ensemble de services indépendants. La séparation des préoccupations est l'une des propriétés fondamentales de l'architecture SOA. Elle permet à chaque service d'être centré sur son propre rôle. De cette façon, les évolutions de l'application sont localisées au sein des services, ce qui permet au reste de l'application de rester stable.

Non seulement ces deux architectures promeuvent l'autonomie, l'indépendance, la modularité et l'accessibilité de leurs agents et de leurs services, mais aussi un couplage étroit pour garantir la communication entre deux ou plusieurs agents et deux ou plusieurs services. Le couplage étroit entre agents ou services selon l'architecture utilisée, est, d'une part, le témoin que l'intégrité, à savoir, l'hétérogénéité des systèmes est respectée, et d'autre part, une propriété indispensable pour permettre la modularité d'une architecture et tendre vers une interopérabilité *plug and play* entre les systèmes.

Depuis plusieurs années maintenant, l'architecture *Arrowhead* [49] s'est imposée comme une des architectures de référence pour l'intégration et l'interopérabilité des IoT, afin d'assouvir la vision qui

émane des systèmes CPS, et plus généralement, de l'Industrie 4.0 [98]. Basés sur les principes de l'architecture SOA, les systèmes sont décomposés en services capables de fournir et de consommer de l'information provenant d'autres services. De ce fait, l'échange d'informations se produit entre un système fournisseur de services (ou *service producer*S_p) et un système consommateur de services (ou *service consumer*S_c). La modularité, ou la composition de services, permet ainsi d'accroître la collaboration entre les systèmes par l'intermédiaire de leurs services, afin de répondre à des activités complexes qu'un système seul ne pourrait résoudre individuellement. Cependant, la résolution d'activités complexes exige que les systèmes participants puissent échanger des informations. L'un des principaux défis consiste à assurer l'interopérabilité entre les IoT et les systèmes au niveau des services, en fournissant un échange d'informations indépendamment des protocoles de communication, de l'encodage des données et de la sémantique des données [134]. Des méthodes et des technologies ont été proposées pour garantir la communication entre deux systèmes hétérogènes par l'intermédiaire d'adaptateurs [134] ou de traducteurs [50, 169]. L'objectif d'un service de traduction est de procurer une conversion au niveau du protocole de communication, de l'encodage des données et de la sémantique des données, entre un service producteur de données S_p et un service consommateur de données S_c qui, de base, ne sont pas conçus pour interagir.

Cependant, les études actuelles portent leur attention sur des adaptateurs capables d'exécuter des traductions au niveau des protocoles de communication et de l'encodage des données, sans pour autant apporter de véritables solutions aux problèmes d'hétérogénéité sémantique. Dans la suite de ce chapitre, un service propre à la traduction sémantique des données sera présenté. Il intégrera évidement la proposition édifiée au Chapitre 3, qui sera déclinée en deux modules : (1) Le premier permettant la construction d'un traducteur automatique de données grâce à l'apprentissage par renforcement des modèles de transformation ; (2) Le second utilisant le traducteur précédemment appris pour assurer la fonction d'intéropérabilité sémantique en traduisant les messages échangés entre un S_p et un S_c. Étant donnée que l'interopérabilité est dite "dirigée par les modèles", des services spécifiques permettant l'injection des données au sein d'un espace technique des modèles, ainsi que leur extraction vers l'espace technique des systèmes, ont été développés et seront présentés.

5.1.1 Définition d'une application et de ses services

Pour commencer, il est important de clarifier les composants utilisés par une architecture orientée service.

Une application. Une application SOA est constituée de plusieurs services interconnectés qui coopèrent pour fournir des fonctionnalités complexes. Du point de vue métier, il s'agit d'un programme conçu pour répondre à des besoins très spécifiques en réalisant des fonctions et des opérations qui dépendent des contraintes et de l'environnement du métier en question.

Un service. Un service représente une unité fonctionnelle, autonome et réutilisable d'un système, qui expose une ou plusieurs fonctionnalités spécifiques, via des interfaces bien définies qui permettent d'être utilisées par un autre système. Du point de vue métier, il s'agit d'une fonction ou d'une opération très spécifique qui encapsule une logique métier.

Un système. Un système, qu'il appartienne à la grande famille des IoT , ou des systèmes de production tels que les systèmes d'informations et de conception, est un objet physique ou logiciel spécifique à un métier, ou a un domaine, qui procure des fonctionnalités sous forme de services S_p ou S_c.

En résumé, une application exploite les fonctionnalités de plusieurs systèmes par l'intermédiaire des services qu'ils exposent, pour réaliser un but bien précis. En un sens, une application est elle-même un système de systèmes, et plus globalement, un système dit complexe. La Figure 5.1 présente la structure générale, sous forme d'un diagramme de classes, d'une application *Apps*. Notons que l'intérêt de ce chapitre n'est pas de présenter une méthode de conception d'une architecture SOA, mais de présenter comment le service de traduction (*Transformator service*) peut être utilisé et intégré au sein d'une application orientée services.

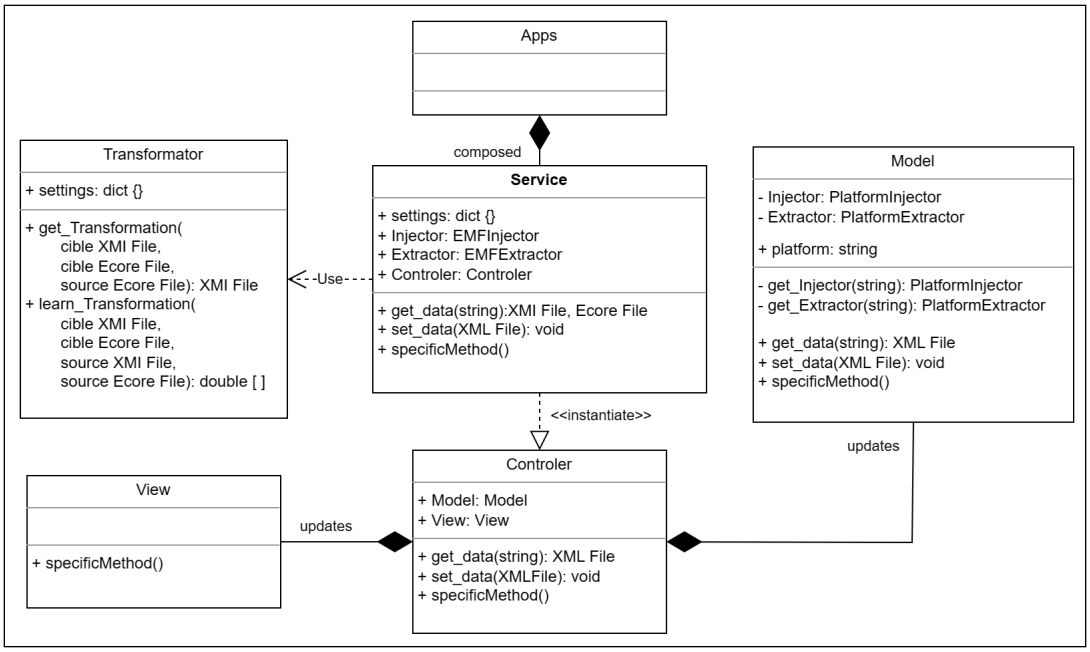


FIGURE 5.1 – Diagramme de classes d'une application

Une application est donc constituée de plusieurs services. Ces services sont définis par un ensemble de méthodes spécifiques qui encapsule la logique métier. Chaque service est construit suivant les principes du *design pattern* Modèle-Vue-Contrôleur (MVC), où :

- Le *Modèle* gère la manipulation et le stockage des données, ainsi que les opérations qui peuvent être effectuées sur ces données. Il comporte donc des méthodes spécifiques pour interagir avec les données. Notons que le modèle est composé d'un injecteur (*injector*) et d'une extracteur (*extractor*) spécifiques à la technologie de la plateforme, qui permet de faire la bascule entre l'injection des données dans l'espace technique des modèles, et l'extraction des données depuis l'espace technique des modèles, vers l'espace technique des systèmes ;
- La *Vue* est responsable de l'affichage des données au sein d'une interface utilisateur interactive

qui présente les informations dans un format compréhensible par l'utilisateur. Dans le cadre de l'architecture SOA présentée, la vue n'est pas nécessairement utilisée ;

– Le *Contrôleur* agit comme intermédiaire entre le modèle, la vue et le service. Il reçoit des demandes provenant de la vue, ou du service, et interagit directement avec le modèle pour répondre à la requête (récupération de données, mise à jour des données...). Dans le cas où le service jouerait le rôle de service S_c, les données reçues seraient intégrées ou mises à jour dans le modèle, tandis que dans le cas d'un service S_p, les contrôleurs exécuteraient une requête sur le modèle pour récupérer, puis transférer, les données demandées par un service S_c.

L'architecture MVC est très efficace lorsqu'il s'agit d'implémenter de nouvelles fonctionnalités sans pour autant affecter l'intégralité de l'architecture.

Enfin, lors d'un échange d'informations entre deux services, le service S_c peut utiliser le service de transformation soit, pour traduire le message reçu dans une sémantique interprétable, soit, pour apprendre les relations sémantiques existantes entre la structure des données internes du service qui vient de produire les données. Dans le second cas, la création du mappage est dite "à la volée", pour permettre un couplage dynamique entre deux ou plusieurs services [164].

5.1.2 Proposition : une architecture orientée service dirigée par les modèles

A présent, nous allons nous intéresser de plus près au service de transformation, qui est en charge de la traduction du message échangé entre un service S_p et un service S_c.communication entre deux services.

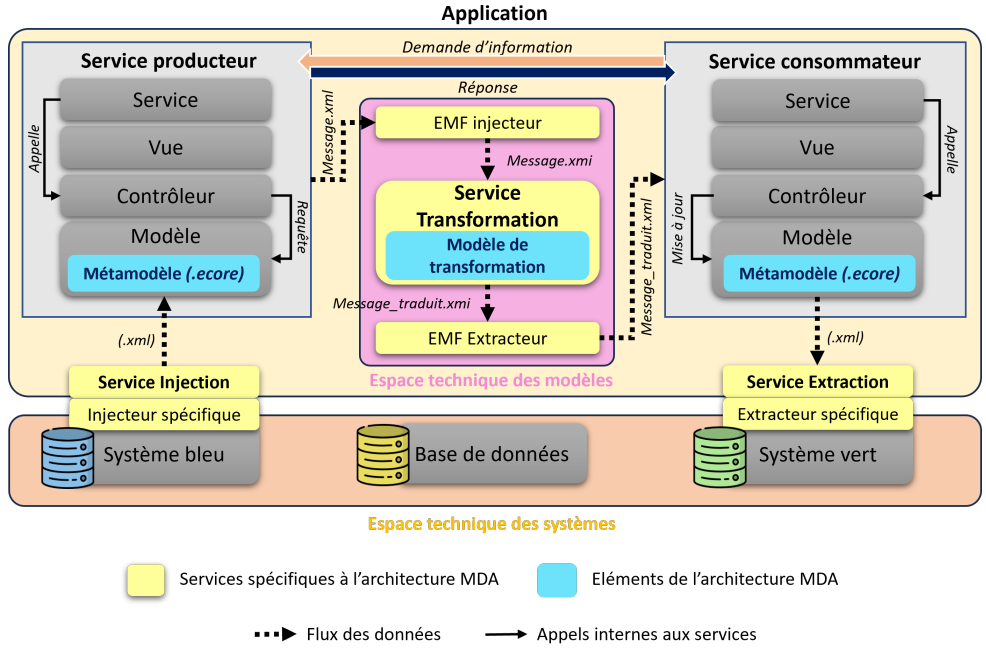


FIGURE 5.2 – Processus de communication entre deux services avec traduction sémantique du message échangé

La Figure 5.2 illustre l'ensemble du processus de communication et de transformation qui fait intervenir les différents services et éléments propres à l'architecture MDA, à savoir les services d'injection, d'extraction et de transformation. Le processus de communication peut être découpé en trois étapes : (1) L'injection des données du service S_p, par l'intermédiaire d'un injecteur spécifique au système, au sein de l'espace technique des modèles ; (2) La transformation, ou la traduction du message émis dans la sémantique du service S_p vers la sémantique du service S_c pour permettre son interprétation ; (3) L'extraction des données, après traduction du message, à partir de l'espace technique des modèles vers l'espace technique des systèmes par l'intermédiaire d'une service d'extraction spécifique au système. Le diagramme de séquence de la Figure 5.3 reprend l'ensemble du processus de transformation.

5.1.2.1 Etape 1 : Processus d'injection

Pour commencer, un service S_c soumet une demande d'informations à un service qui va produire les données souhaitées, i.e. le service S_p. Pour répondre à la requête émise par le service S_c, le contrôleur du service S_p interroge, par l'intermédiaire du modèle, le système bleu où sont stockées les données. La récupération des données est réalisée par un injecteur spécifique au système bleu qui injecte les données désirées dans un fichier au format XML, avant d'être lui-même injecté dans l'espace technique des modèles (voir Figure 5.3 ref : "processus d'injection spécifique à la plateforme").

L'injecteur spécifique utilise la technologie propre au système bleu pour interagir avec lui. Le diagramme de classes de la Figure 5.4 reprend sous les traits du *factory design pattern*, l'instanciation d'un injecteur spécifique (*PltaformInjectorA* ou *PltaformInjectorB* par exemple) en fonction de la technologie employée par le système (telles que les technologies Microsoft Excel, Access et word par exemple). L'utilisation d'un pattern de type *Factory* permet, à partir de l'interface *Injector*, d'instancier un injecteur spécifique selon la technologie du système afin de garantir un accès à ses données. Une collection d'injecteurs peut alors être implémentée dynamiquement selon les systèmes à ajouter, à retirer ou à modifier, dans le cas d'un changement de technologie. Par conséquent, lorsque l'application emploie un nouveau système par l'intermédiaire de ses services, l'injecteur spécifique au système doit être implémenté. Ce principe d'implémentation apporte à une architecture logicielle davantage de modularité et de flexibilité, et à pour conséquence d'améliorer l'intégration et l'interopérabilité des systèmes au sein de l'application, tout en minimisant l'intervention d'experts en charge de construire les différents connecteurs entre les systèmes.

Etant donné que la traduction du message opère dans l'espace technique des modèles, il est alors nécessaire d'instancier l'injecteur spécifique à la plateforme *Eclipse Modeling Framework* (EMF). L'injecteur spécifique EMF permet le passage des données depuis l'espace technique des systèmes au format XML, vers l'espace technique des modèles au format XMI. Le diagramme de séquence du processus d'injection des données dans l'espace technique des modèles, présenté en Figure 5.5, peut être résumé en trois étapes : (1) L'extraction du schéma du fichier XML, ou XML Schema Definition (XSD), qui donne la structuration et les contraintes sous-jacentes aux données ; (2) La conversion du fichier XSD au format Ecore ; (3) L'injection des données au format XML au sein de EMF dans le format XMI, lisible par le service de transformation.

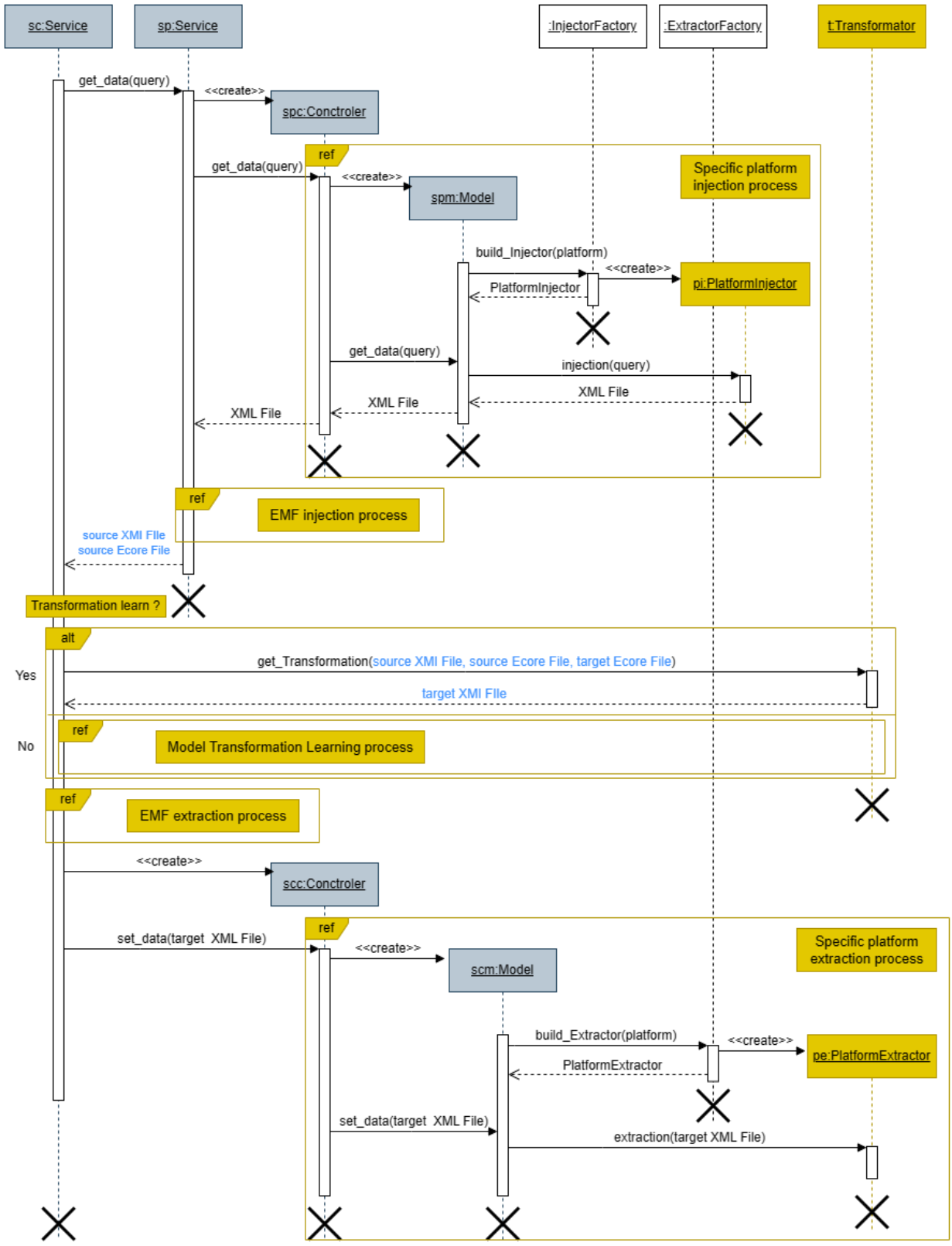


FIGURE 5.3 – Diagramme de séquence du processus de communication entre deux services

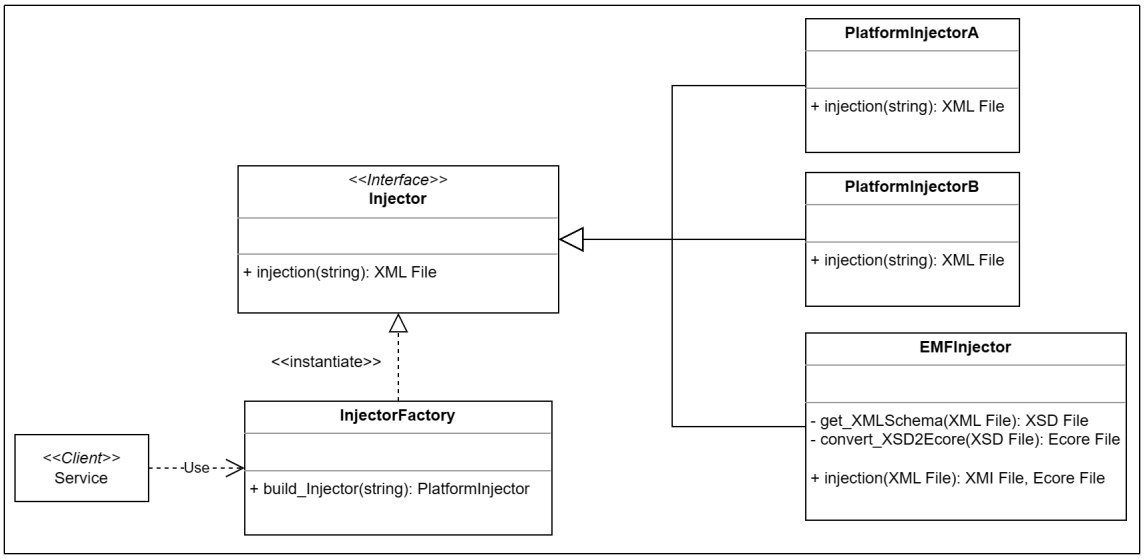


FIGURE 5.4 – Diagramme de classe du service d'injection

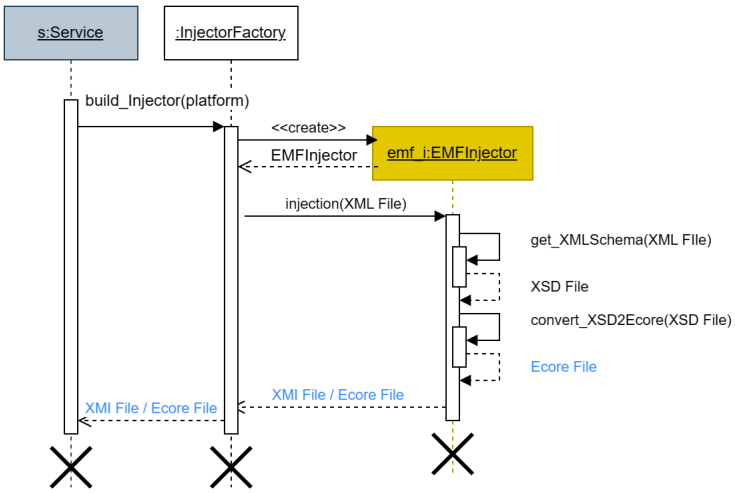


FIGURE 5.5 – Diagramme de séquence du processus d'injection des données dans l'espace technique des modèles *Eclipse Modeling Framework*

5.1.2.2 Etape 2 : Processus de transformation

Une fois le message reçu par le service consommateur, les données portées par le message seront traduites dans une sémantique et une structure interprétable par le S_c grâce au service de transformation.

Étant donné que l'approche proposé repose sur les principes d'une architecture fédérée, chaque service dispose de son propre métamodèle interne, qui décrit la structure et la sémantique de ses données [165]. La création de mappage entre deux métamodèles permet d'instaurer un couplage lâche entre

deux services, et par extension d'assurer l'interopérabilité entre deux systèmes. Le service de transformation est en charge d'inférer, à la volée, le mappage entre deux ou plusieurs métamodèles, ainsi que de ré-exploiter le mappage précédemment édité dans le cas d'une transformation, ou traduction de données. Le service de transformation intervient alors selon deux scénarios :

Le premier scénario intervient dans le cas où la transformation entre le métamodèle du service S_p et le métamodèle du service S_c est déjà connue. Le service de transformation exploite alors le modèle de transformation, à savoir, la matrice de transformation inférée au paravent, pour traduire le message dans la sémantique des données du service S_c. Une fois la transformation réalisée, le message traduit, au format XMI, sera conforme au métamodèle du service S_c et pourra donc être exploité sans ambiguïté.

En revanche, dans le scénario où un système vient d'être ajouté à l'application, la communication entre les services dont il est propriétaire et les autres services de l'application doit être construite à la volée. Une interopérabilité établie à la volée fait référence à la capacité des systèmes à échanger des données de manière flexible et adaptative, sans aucune configuration préalable. Concernant le service de transformation, cela signifie que le mappage entre les métamodèles doit être construit automatiquement et dynamiquement en temps réel [164].

La construction du modèle de transformation spécifique à la communication entre un service S_p et un service S_c (voir Figure 5.3 ref : "processus d'apprentissage du modèle de transformation"), suit le processus d'apprentissage des modèles de transformation présenté en Chapitre 3.

5.1.2.3 Etape 3 : Processus d'extraction

Une fois le message transmis par le service S_p, puis traduit par le service de traduction dans la sémantique et la structure des données du métamodèle du service S_c, il est nécessaire, dans un premier temps, d'extraire les données de l'espace technique des modèles EMF vers l'espace technique des systèmes, puis, dans un second temps, d'intégrer les données dans dans le système vert à l'aide d'un extracteur spécifique à la technologie du système vert.

Le service d'extraction joue le rôle d'intermédiaire entre l'espace technique des modèles et l'espace technique des systèmes. A l'image du service d'injection, le service d'extraction a été conçu suivant les principes du *factory design pattern* (voir Figure 5.7). De ce fait, un extracteur spécifique à la technologie du système (*PltaformExtracorA* ou *PltaformExtractorB* par exemple) est instancié par l'intermédiaire d'interface *Extractor*. Grâce au service d'extraction, les données précédemment traduites sont intégrées dans le système vert.

Le diagramme de séquence du processus d'extraction des données dans l'espace technique des systèmes, présenté en Figure 5.5, permet de transformer un fichier au format XMI exploitable dans l'espace technique des modèles, en fichier au format XML exploitable par le système cible,en l'occurrence, le système vert dans l'exemple utilisé.

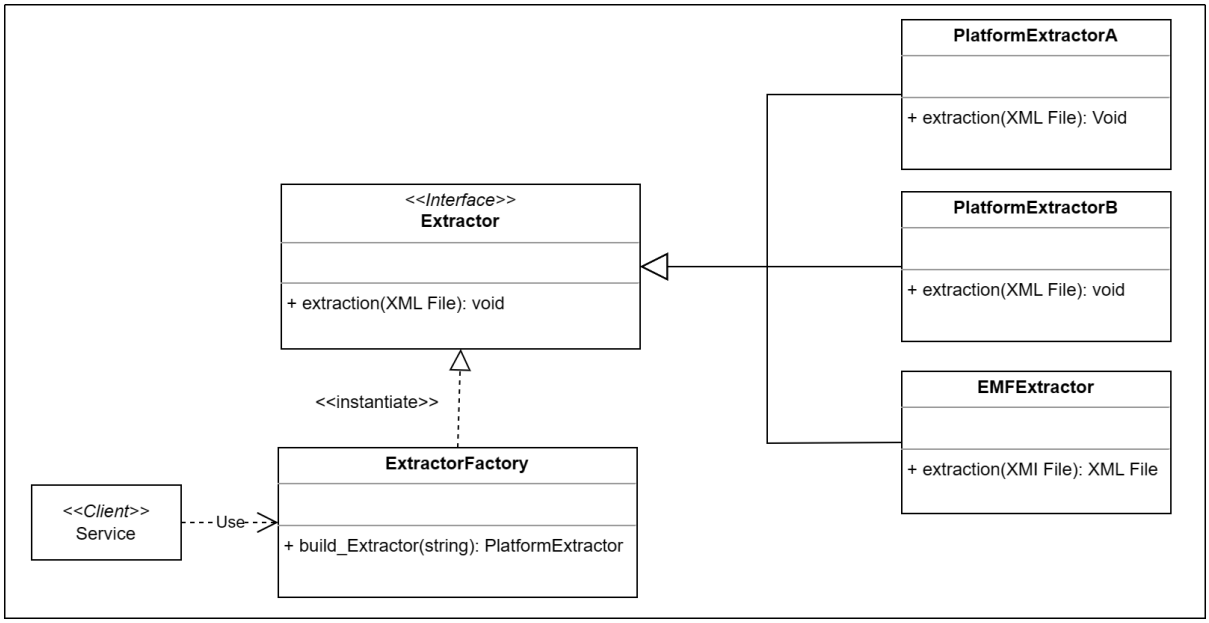


FIGURE 5.6 – Diagramme de classes du service d'extraction

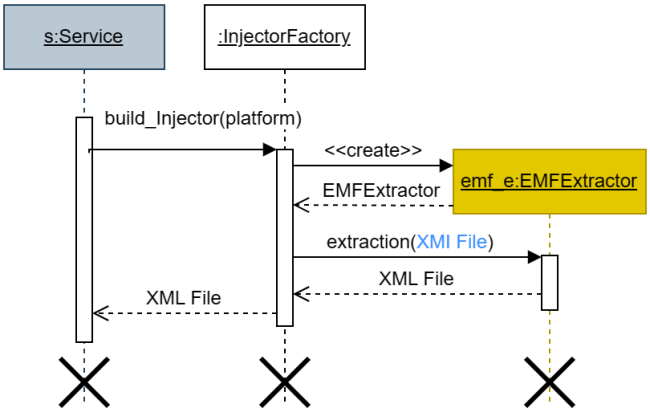


FIGURE 5.7 – Diagramme de séquence du processus d'extraction depuis l'espace technique des modèles *Eclipse Modeling Framework*

5.1.3 Documentation de l'architecture

La structure d'une architecture SOA doit être correctement documentée, de sorte à faciliter l'intégration de nouvelles sources d'informations. Par exemple, l'architecture *Arrowhead* est divisée en trois niveaux de documentation qui procurent une vue abstraite et instanciable d'un système de système, d'un système et d'un service [134].

La structure documentaire de l'architecture SOA dirigée par les modèles pour implémenter un service de transformation suit deux niveaux (voir Figure 5.8) : (1) Les *Apps*, à savoir, les applications

spécifiques qui exploitent des services pour accomplir des tâches bien précises ; (2) Le services, qui encapsulent la fonctionnalité d'un système, ou la logique d'un métier. Le niveau propre aux services est lui-même découpé en trois sous-niveaux : (1) Les services appelés *core*, propres à l'architecture MDA, à savoir les services d'injection, d'extraction et de transformation ; (2) Les services consommateurs de données ; (3) Les services producteurs de données. Les services consommateurs et producteurs sont eux-même composés d'un Modèle, d'une Vue et d'un Contrôleur. Le fichier *Settings*, à l'image du service de description de l'architecture *Arrowhead* [20], spécifie le type de système exploité par le service et le protocole de communication ainsi que l'encodage des données utilisé par le service.

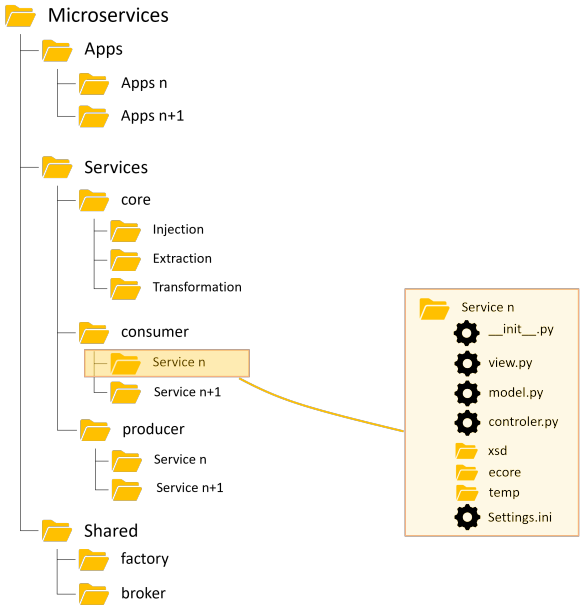


FIGURE 5.8 – Structure de la documentation d'une architecture SOA dirigée par les modèles

5.2 Cas d'étude

L'industrie connaît des mutations exceptionnelles avec la digitalisation. Le challenge est multi-forme : il touche autant au modèle des entreprises, qu'à leur méthode de production et leur organisation. Pour répondre efficacement sur son marché, le tournant du numérique et les transformations majeures qui l'accompagnent semble inévitable.

Dans un tel environnement, la compétitivité d'une entreprise dépend de plus en plus de sa flexibilité et de sa capacité à innover. C'est pourquoi *Thalès Alenia Space* (TAS) entame sa transformation digitale. L'objectif ? Optimiser le processus de production en produisant mieux, à de meilleurs coûts, tout en offrant de nouveaux produits/services grâce à des systèmes intelligents permettant de collecter, stocker, gérer et analyser les données.

Les cas d'études présentés dans cette section ont été mis en place pendant les phases d'Assemblages, Intégrations et Tests des Satellites (AIT). Avant de présenter chacun des cas d'étude, les sous-sections

suivantes introduiront le contexte industriel global de l'AIT (voir Section 5.2.1), ainsi que sa stratégie en terme d'innovation et de digitalisation (voir Section 5.2.2.

5.2.1 Contexte général de l'AIT

Les phases de l'AIT constituent les derniers jalons dans la vie du satellite. L'Assemblage et l'Intégration consistent à réaliser, à partir de sous-systèmes (structures, équipements, câblages) un système (le satellite). Les phases de tests sont réalisées après les phases d'assemblages en environnement contrôlé (ISO8 ou ISO5), en environnement thermique contrôlé (caisson) et en environnement mécanique (pot virant et chambre acoustique). Le département de l'AIT a pour mission :

- L'assemblage, l'intégration et la réalisation des tests au niveau charge utile et satellite.
- La livraison du satellite avec des résultats d'essais démontrant la conformité du satellite au regard des performances attendues.
- La préparation de la campagne de lancement du satellite ainsi que les opérations jusqu'au décollage

L'AIT est un processus constitué de plusieurs phases (voir Figure 5.9 :

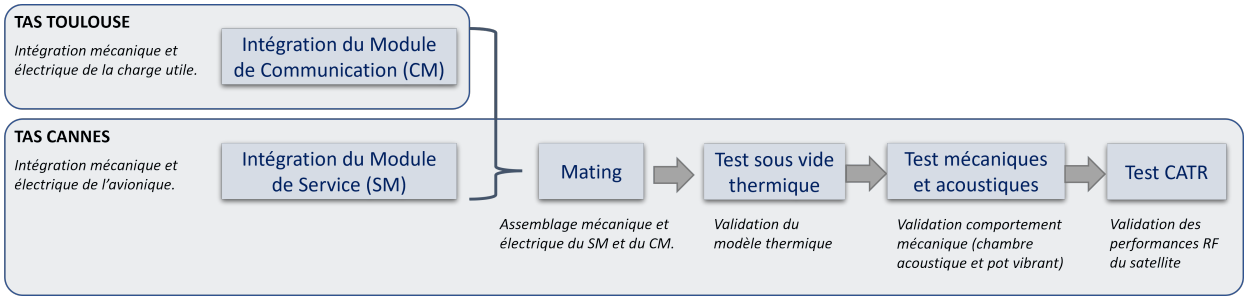


FIGURE 5.9 – Processus de l'AIT Satellite

- *Intégration du Module de Communication* (CM) réalisé à Toulouse : intégration mécanique et électrique de la charge utile. La charge utile d'un satellite correspond à la partie de l'engin spatial qui est destinée à remplir les objectifs de la mission.
- *Intégration du Module de Service* (SM) réalisé à Cannes : intégration mécanique et électrique de l'avionique.
- *Mating* : assemblage mécanique et électrique du SM et du CM.
- *Vide thermique en caisson de 500 m3* : test de balance permettant de valider le modèle thermique, et test de thermique permettant de vérifier les performances du satellite au palier chaud et froid dans le vide.
- *Tests mécaniques vibrations (pot vibrant) et acoustiques (chambre acoustique)* : permettent de vérifier le comportement mécanique du satellite lorsqu'il est soumis à un environnement similaire à celui qu'il rencontrera lors du décollage du lanceur.
- *CATR (chambre anchoïade)* : permet de valider les performances Radio Fréquence du satellite en rayonné.

5.2.2 Stratégie de l'AIT

Face à l'évolution constante des besoins clients, à leurs exigences croissantes et à la montée de la concurrence, l'AIT développe une organisation capable de s'adapter en permanence aux besoins du client.

Pour y parvenir, le département AIT encourage les initiatives dans l'amélioration du processus, dans la résolution des problèmes et dans l'amélioration des conditions de travail, dans le but de créer un avantage concurrentiel. La transformation digitale est au coeur de ces grands changements. Des axes d'innovation sont clairement identifiés et portent sur la création d'infrastructures pour la communication des IoT, le déploiement d'applications intelligentes en capacité d'optimiser la planification des activités pendant les phases de l'AIT, ou encore l'intégration d'outils de réalité augmentée pour fournir les instructions de montage de panneaux lors de l'assemblage. La notion d'interopérabilité des systèmes est intrinsèque aux axes d'innovation prédéfinis. L'échange d'informations et la coopération des systèmes pour simplifier la prise de décision dans un environnement complexe tel que l'AIT est primordial.

Les deux prochaines sections illustreront précisément l'intérêt d'exploiter un mécanisme de connexion générique, automatique et dynamique, grâce à l'apprentissage des modèles de transformation, pour assurer l'interopérabilité des systèmes dans un contexte industriel tel que celui de l'AIT. Deux cas d'étude seront présentés : l'automatisation des documents de type *As-Built* (voir Section 5.2.3), et l'automatisation de la planification des activités en AIT

5.2.3 Cas d'étude n°1 : L'automatisation de la conception des documents *As-Built*

5.2.3.1 Description du contexte

Contexte. Une *As-Built* (conforme à l'exécution), consiste à documenter l'encours des activités d'assemblage d'un satellite. Ce document permet de valider la conformité des opérations d'assemblages et d'intégrations selon les spécifications initiales. Les documents *As-Built* sont généralement édités à la fin de la phase d'intégration du module de service et de la phase de *Mating*, une fois que toutes les modifications et les ajustements par rapport aux plans initiaux ont été pris en compte. Ces documents fournissent une représentation précise de l'encours des activités d'assemblages et d'intégrations, telles que réalisées dans la réalité, par opposition aux plans théoriques initiaux.

Le diagramme BPMN de la Figure 5.10 reprend l'ensemble de processus de création d'une *As-Built* et présente les métiers qui contribuent à la réalisation du document, ainsi que les systèmes utilisés dont sont issues les données.

Le métier d'ordonnanceur est en charge de la réalisation du document *As-Built*. Le service d'ordonnancement des affaires en AIT s'assure en permanence de la disponibilité du matériel nécessaire tout au long du processus de l'AIT Satellite. Il gère également le lancement et la gestion des Ordres de Fabrication (OF), ainsi que le suivi de la planification des besoins. Actuellement, l'ordonnanceur dédie 15 heures de son devis initial de 402 heures par satellite, soit 30 heures au total pour deux *As-Built*.

Problème. Le processus de création d'une *As-Built* est long et fastidieux. Il nécessite de jongler

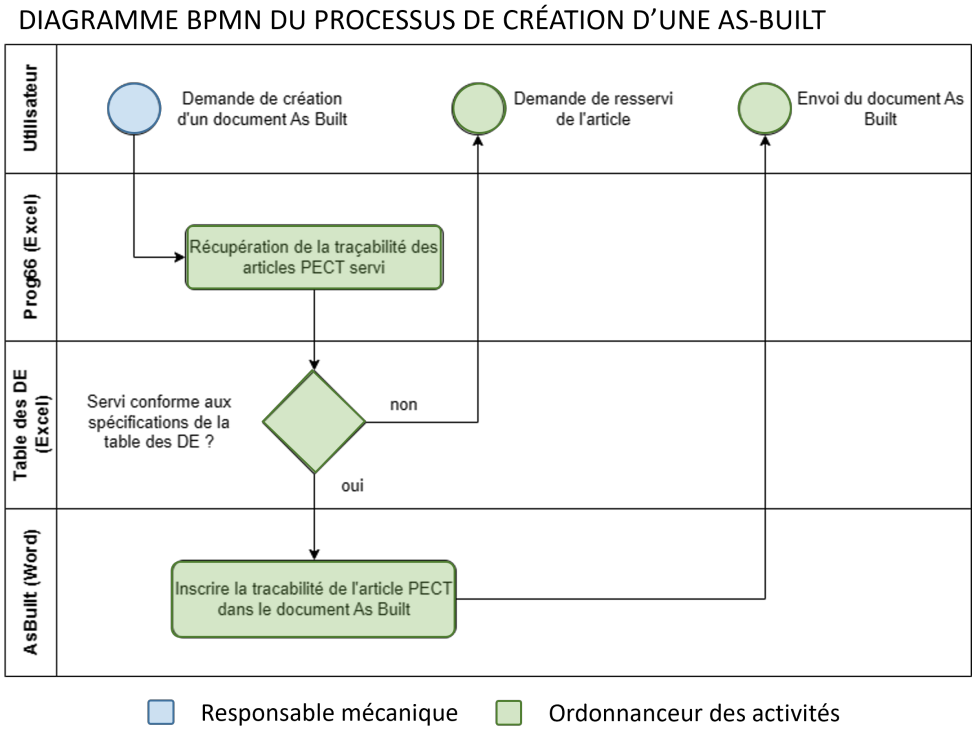


FIGURE 5.10 – Diagramme BPMN du processus de création d'une *As-Built*

entre différentes sources d'informations afin de corréler leurs données. Les différents systèmes utilisés sont les suivants :

- L'ERP de l'usine est en charge du traitement des données relatives à la production (plan de production, gestion des stocks, traçabilité...). Pour construire un document *As-Built*, la récupération de la traçabilité des articles montés sur le satellite est nécessaire pour obtenir une cartographie du réel, à savoir, de l'ensemble des sous-systèmes intégrés et assemblés sur le satellite.
- Le progiciel 3IT contient les gammes et les nomenclatures de conception des satellites. Il intègre également un fichier *Excel* appelé Table DE qui contient l'ensemble des spécifications à respecter lors des opérations d'intégrations et d'assemblages. Par exemple, le fichier Table DE spécifie quel article, selon son numéro de série, doit être monté sur tel repère topologique (emplacement) sur le satellite.
- L'*As-Built* est un document au format *Word*, qui recense la traçabilité des articles montés sur le satellite. En d'autres termes, un *As-Built* recense, pour une liste d'articles donnée, la traçabilité d'un article selon son numéro de série et l'emplacement topologique où il a été monté.

La création d'un document *As-Built* est nécessaire, à la fois, pour archiver la traçabilité des articles montés sur le satellite, et pour identifier d'éventuelles erreurs ou incohérences lors des phases d'intégration et d'assemblage, qui pourraient nécessiter des ajustements ou des réparations. Par exemple, l'intégration d'un élément sur le satellite doit respecter une condition particulière spécifiée par la Table

DE, à savoir, un article, identifié par son numéro de série, doit être monté sur un repère topologique spécifique au satellite. Par conséquent, pour chaque article de l'*As-Built*, l'ordonnanceur doit vérifier que le couple numéro de série, repère topologique est conforme aux spécifications de la Table DE. La corrélation entre les données provenant de l'ERP et du progiciel 3IT doit donc être réalisée pour éditer le document *As-Built*.

Automatiser l'édition d'un document *As-Built* est une activité ardue. Etant donné que la structure des données internes du fichier Table DE varie selon les utilisateurs, la construction d'un mécanisme de récupération des données est une tâche délicate qui nécessiterait la création d'une passerelle unique à chaque fichier. Par ailleurs, durant la vie du fichier Table DE, sa structure interne peut vouée à être modifiée, rendant tout mécanisme de récupération des données obsolète.

Proposition. Afin d'automatiser l'ensemble du processus de création des documents *As-Built*, une approche orientée services exploitant les principes de l'architecture dirigée par les modèles, telle que présenté en Section 5.1.2, sera utilisée. La communication entre les différents services est réalisée par l'intermédiaire du service de transformation qui sera en charge de construire des connecteurs sur mesure entre les différents fichiers. Dans le cas présent, deux connecteurs seront appris par le mécanisme de transformation : un premier connecteur entre le fichier progg66 (extraction Excel de l'ERP) et le fichier Word *As-Built* et un second connecteur entre le fichier Excel Table DE et le fichier Word *As-Built*.

Concernant le connecteur qui relie le fichier prog66 et le fichier word *As-Built*, il permettra la récupération de l'ensemble de la traçabilité des articles articles pour une liste prédéfinie. Le service de transformation sera en charge d'apprendre les relations sémantiques qui lient ces deux fichiers (voir Figure 5.11).

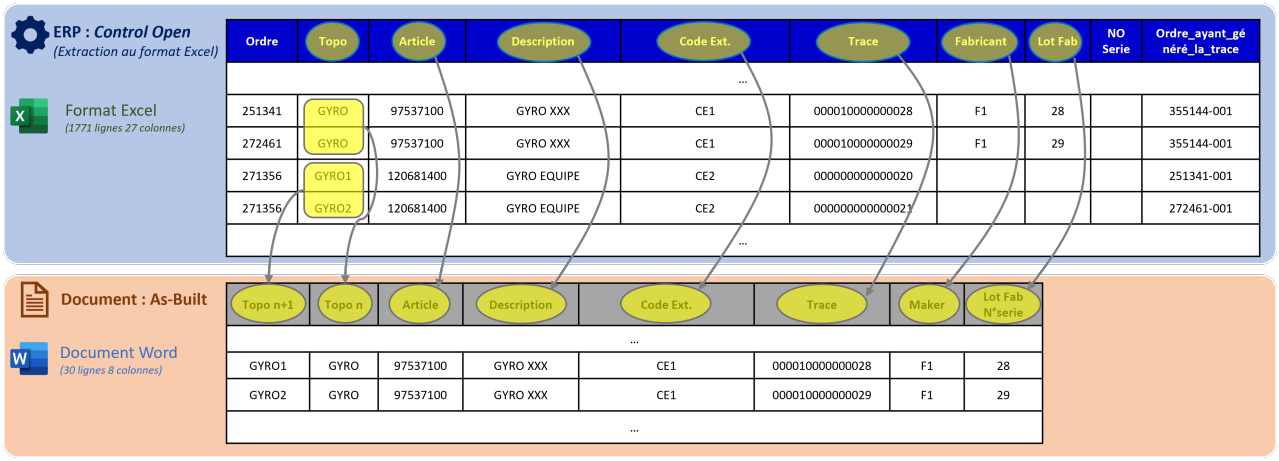


FIGURE 5.11 – Relations d'équivalence entre le fichier Excel extrait de l'ERP et le document *As-Built* au format word

Enfin, le connecteur qui relie le fichier Excel Table DE et le fichier Word *As-Built* permettra de récupérer, pour un article, selon son numéro de série, le repère topologique auquel il est associé. La Figure 5.12 illustre les différentes relations sémantiques que devra apprendre le service de transformation.

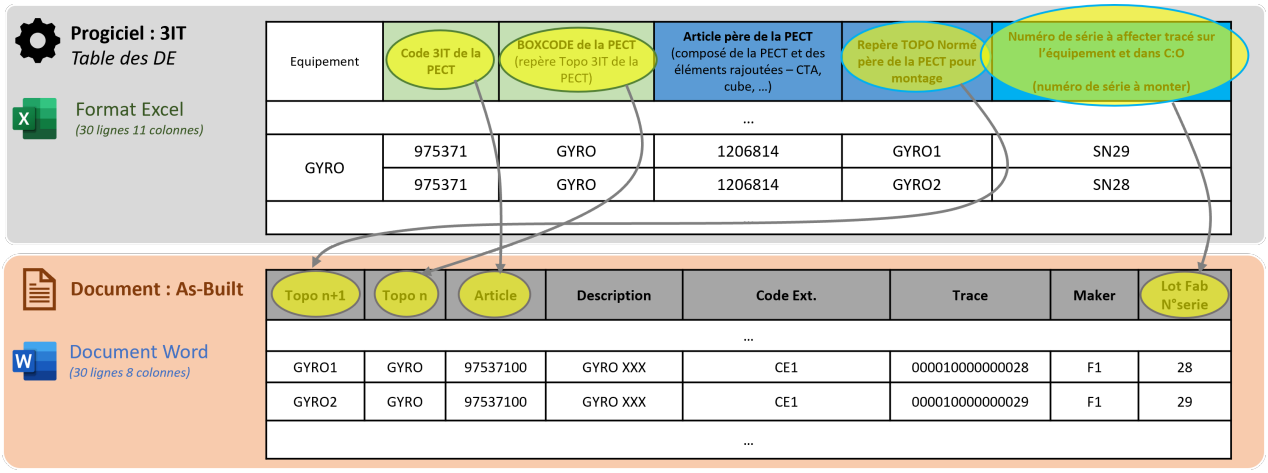


FIGURE 5.12 – Relations d'équivalence entre le fichier Excel Table DE extrait du progiciel 3IT et le document *As-Built* au format word

5.2.3.2 Description technique de la solution

La solution technique permettant l'automatisation de la création d'un document *As-Built* suivra les spécifications de l'architecture logicielle d'une application présentée en Section 5.1.2. L'objectif de l'application est d'automatiser l'édition d'un document *As-Built* grâce à : (1) La récupération automatique de la traçabilité des articles ; (2) La validation des spécifications pour chaque article. Pour cela, l'application sera composée de trois services :

- *Le service producteur prog66* sera en charge de produire les données demandées, à savoir, de fournir la traçabilité des articles demandés. La méthode *get_tracability()*, prenant en paramètre une liste d'articles, sera utilisée pour interroger le fichier *Excel prog66*, puis, récupérer les informations de traçabilité. Evidemment, un mécanisme d'injection propre à la plateforme *Excel* a été développé pour permettre au service *prog66* d'interagir avec le fichier *prog66*.
- *Le service producteur Table DE* sera, quant à lui, en charge de fournir les spécifications d'assemblage des articles, selon leur numéro de série. La méthode *get_specification()*, prenant en paramètre une liste d'articles et leur numéro de série, sera utilisé pour interroger le fichier *Excel Table DE*, puis, récupérer les informations concernant le repère topologique des articles en fonction de leurs numéros de série.
- *Le service consommateur As-Built* sera en charge de récupérer et de traiter les données issues des services *prog66* et *Table DE*. Dans un premier temps, la méthode *check_conformity()* permettra de valider que les spécifications d'assemblage ont été respectées, puis, dans un second temps, la méthode *build_asbuilt()* permettra d'éditer un document *As-Built* de type *Word*. Evidemment, un extracteur spécifique à la plateforme *Word* a été développé pour permettre au service *As-Built* d'insérer les données dans un document *Word*.

Le diagramme BPMN de la Figure 5.13 présente le processus d'orchestration des services *prog66*, *Table DE* et *As-Built*, afin d'automatiser l'édition d'un document *As-Built*.

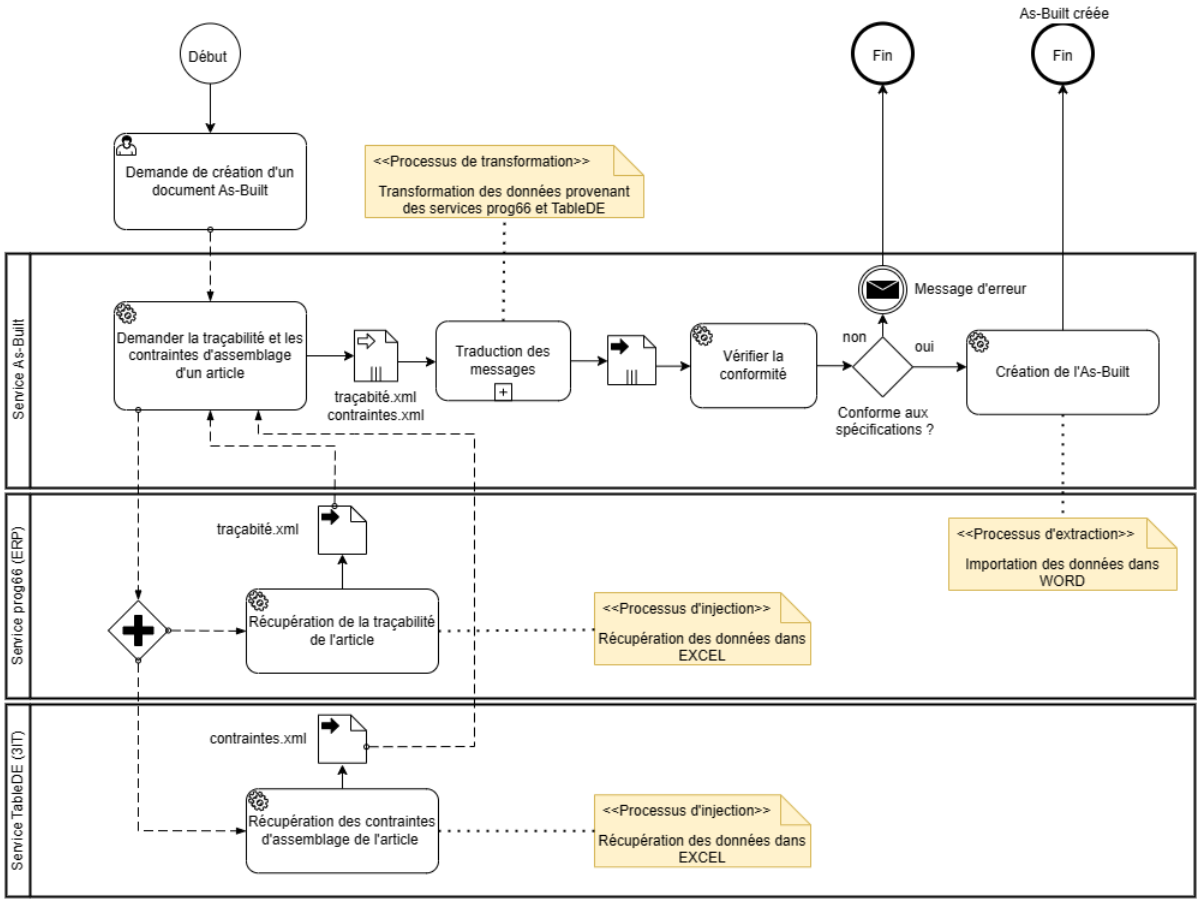


FIGURE 5.13 – Diagramme BPMN du processus de construction d'un document *As-Built*

Le service *As-Built* initialise une communication parallèle avec les services *prog66* et *Table DE* en demandant la traçabilité et les contraintes d'assemblage d'une liste d'articles. La récupération de la traçabilité des articles par le service *prog66*, ainsi que la récupération des contraintes d'assemblage des articles par le service *Table DE* est réalisée par l'intermédiaire d'un injecteur propre à *Excel*. Une fois les données reçues, le service *As-Built* les transmets au service de transformation qui se charge, soit, d'apprendre la transformation de modèle à la volée, soit de traduire les données dans la sémantique interne du service *As-Built*. La traduction des données provenant des services *prog66* et *Table DE* permettra au service *As-Built* de les exploiter sans ambiguïté, afin de vérifier la conformité des assemblages puis, dans le cas où les spécifications sont respectées, d'éditer le document *As-Built*. Si les spécifications d'assemblage ne sont pas respectées, alors un message d'erreur est retourné à l'utilisateur. L'insertion des données dans le document *As-Built* est réalisé par l'intermédiaire d'un extracteur spécifique à la plateforme *Word*.

Le diagramme BPMN de la Figure 5.14 illustre le processus interne du service de transformation. Il peut être découpé en trois sous-processus successifs : le processus d'injection, le processus de transformation, le processus d'extraction

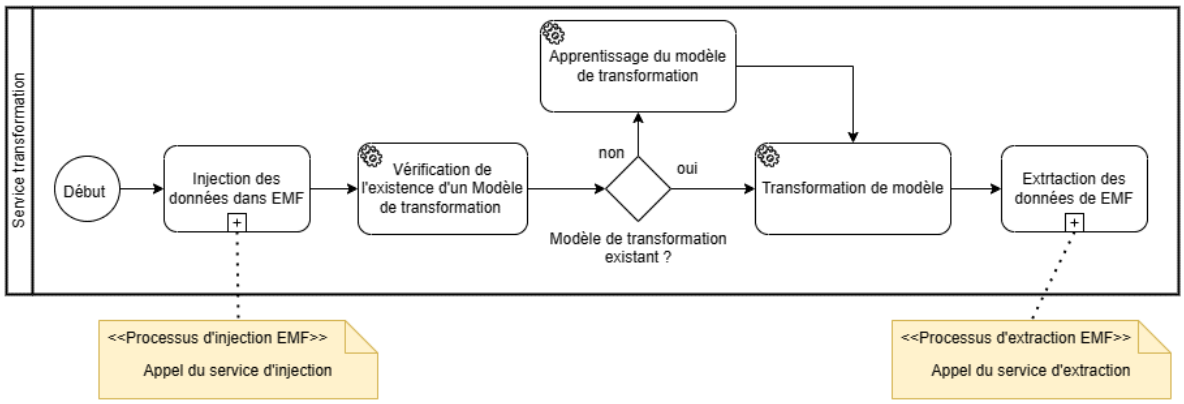


FIGURE 5.14 – Diagramme BPMN du processus de de transformation

Prenons pour exemple le processus de transformation qui a permis d'établir un connecteur sémantique entre les services *prog66* et *AsBuilt*.

Processus d'injection. Une fois la demande de données reçue par le service *prog66*, la première étape consiste à injecter les données provenant de l'espace technique de systèmes, vers l'espace technique des modèles, là où la traduction des données (la transformation des modèles) pourra être réalisée. Dans le cas présent, les plateformes *Excel* et *Word* sont les seuls systèmes à intervenir. L'injecteur propre à la plateforme EMF sera utilisé pour intégrer les données provenant du fichier *prog66* au format XML et les convertir au format XMI. Lors du processus d'injection, le schéma XSD du fichier XML est extrait afin de construire automatiquement le métamodèle du fichier *prog66* au format *Ecore*. La création du métamodèle *prog66* est une étape primordiale pour pouvoir intégrer les données du fichier XML dans un fichier XMI conforme à un métamodèle.

Processus de transformation. A présent que le message émis par le service *prog66* a été reçu par le service *As-Built*, les données qu'il contient vont pouvoir être transformées dans une sémantique comprise par le service *As-Built*, de sorte à permettre leur manipulation. Le service de transformation réutilise la matrice de transformation précédemment apprise pour transformer le modèle *prog66.xmi*, conforme au métamodèle *prog66.ecore*, en un modèle *AsBuilt.xmi*, conforme au métamodèle *AsBuilt.ecore*. Notons que, dans le cas où la matrice de transformation, qui permet le passage d'un modèle conforme au métamodèle *prog66.ecore* vers un modèle conforme au métamodèle *AsBuilt.ecore*, est inconnue, une phase d'apprentissage des règles de transformation est nécessaire. La matrice de transformation est donc créée à la volée pour assurer la communication entre les services *prog66.ecore* et le service *AsBuilt.ecore*. Le processus d'apprentissage des modèles de transformation est décrit en Section 5.1.2.2. C'est d'ailleurs pendant le processus d'apprentissage que le métamodèle *AsBuilt.ecore* est construit à partir du schéma XSD du fichier XML qui provient du fichier *AsBuilt.xml* lors du sous-processus d'injection des données issues de la platefrome *Word*.

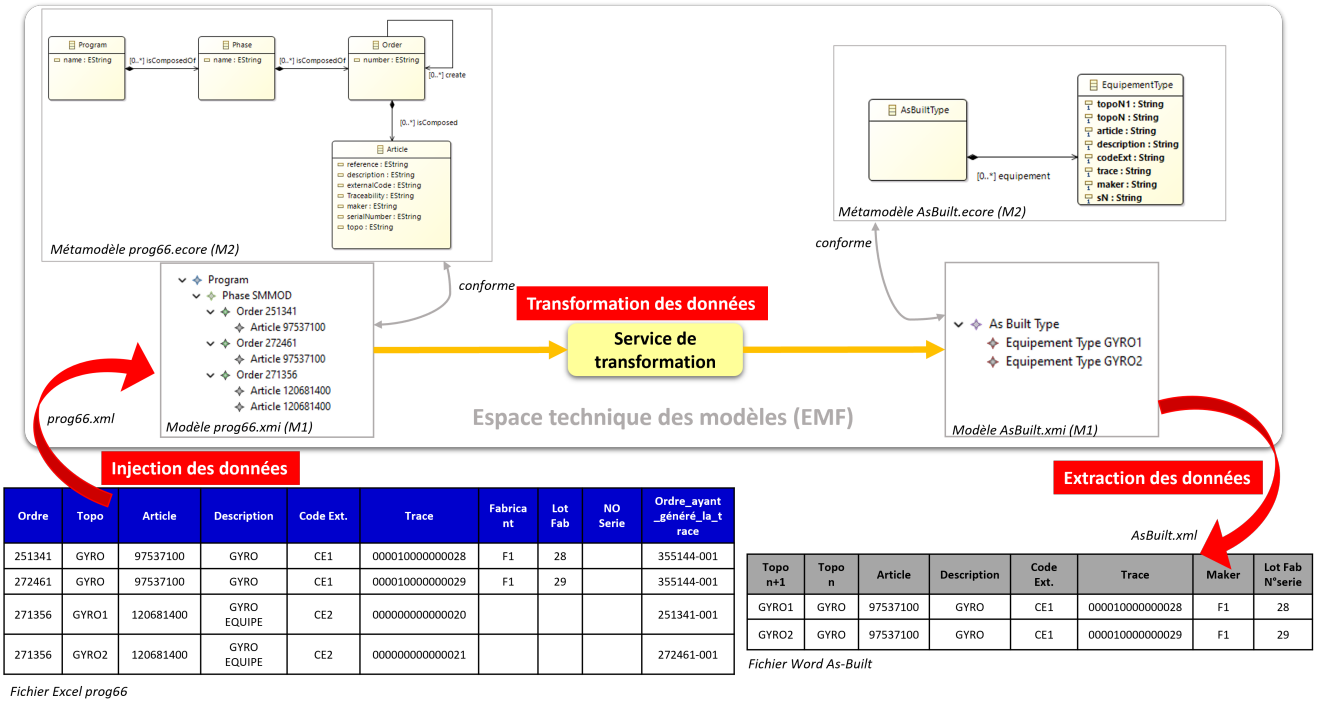
Le Tableau 5.1 recense les patterns de transformation qui interviennent durant les transformations *prog662AsBuilt* et *TableDE2AsBuilt*. Dans le cas présent, la caractérisation de ces deux transformations a permis de mettre en évidence que les patterns de transformation (pattern (c1) et σ_r) à inférer par l'algorithme d'apprentissage sont les mêmes que ceux qui interviennent lors des transformations

	Caractéristiques structurelles							***Caractéristiques terminologiques***			
	Patterns de transformation				*Conditions*						
Transformation	$[1-1]$	$[1-n]$	$[n-1]$	$[n-m]$	σ_c	σ_a	σ_r	C	R	A	V
Families2Persons	-	-	(c2)	-	-	-	x	(e4)	(e4)	(e3)	-
Class2Relational	(c1)	(c5)	-	(c4)	x	x	-	(e6)	(e6)	(e6)	(d2 ;d5)
prog662AsBuilt	(c1)	-	-	-	-	-	x	(e6)	(e6)	(e6)	-
TableDE2AsBuilt	(c1)	-	-	-	-	-	-	(e6)	(e6)	(e6)	-

TABLE 5.1 – Caractérisation des transformations de modèles du cas d'étude n°1

Class2Relational et *Families2Persons*. L'algorithme d'apprentissage ayant déjà été validé pour ces deux patterns, il est alors possible d'émettre l'hypothèse que l'algorithme d'apprentissage du modèle de transformation sera en capacité de dériver l'intégralité des règles de transformations. Dans ce cas bien précis, la caractérisation des transformations de modèles permet d'apporter un niveau de confiance aux résultats de l'algorithme.

La Figure 5.15 présente l'ensemble du processus de transformation appliqué aux données provenant du fichier *prog66*. Les modèles *prog66.xmi* et *AsBuilt.xmi* qui ont été exploités lors de la phase d'apprentissage sont présenté en Annexe B.1 (voir respectivement les figures B.2 et B.4). Afin de simplifier l'apprentissage du modèle de transformation, les données du fichier *prog66* et du fichier *As-Built* ont été filtrées sur les mêmes articles, à savoir les articles GYRO ayant pour référence 97537100.



Ordre	Topo	Article	Description	Code Ext.	Trace	Fabricant	Lot Fab	NO Serie	Ordre_ayant_généré_la_trace
251341	GYRO	97537100	GYRO	CE1	000010000000028	F1	28		355144-001
272461	GYRO	97537100	GYRO	CE1	000010000000029	F1	29		355144-001
271356	GYRO1	120681400	GYRO EQUIPE	CE2	000000000000020				251341-001
271356	GYRO2	120681400	GYRO EQUIPE	CE2	000000000000021				272461-001

Topo n+1	Topo n	Article	Description	Code Ext.	Trace	Maker	Lot Fab N°serie
GYRO1	GYRO	97537100	GYRO	CE1	000010000000028	F1	28
GYRO2	GYRO	97537100	GYRO	CE1	000010000000029	F1	29

FIGURE 5.15 – Processus de transformation des données du prog66

Processus d'extraction. Maintenant que les données ont correctement été transformées et que la conformité de l'assemblage a été vérifiée, les données peuvent être à présent intégrer dans l'*As-Built*. Pour cela, un extracteur propre à la platefrome *Word* permet le passage de l'espace technique des modèles, vers l'espace technique des systèmes.

5.2.3.3 Résultats

L'objectif principal de ce premier cas d'étude est d'automatiser le processus de création des documents *Word* de type *As-Built*. Deux processus sous-jacents ont été automatisés : (1) Le processus de récupération de la traçabilité d'une liste d'articles dans le fichier *Excell* prog66 extrait de l'ERP ; (2) Le processus de vérification des spécifications d'assemblages et d'intégrations pour une liste d'articles dans le fichier *Excell* Table DE extrait de lu progiciel 3IT. Le principal défi consiste à automatiser la communication entre les services producteurs de données *prog66* et *Table DE* et le service *As-Built* qui va consommer ces données. Le Tableau 5.2 présente les résultats obtenus pour l'apprentissage des transformations *prog662AsBuilt* et *TableDE2AsBuilt*.

	Performances d'apprentissage				***Performance de transformation***
Transformation	*Temps (s)*	*P*	*R*	*F*	*F*
prog662AsBuilt	0.55	1.0	1.0	1.0	1.0
TableDE2AsBuilt	0.68	1.0	1.0	1.0	1.0

TABLE 5.2 – Résultats de l'apprentissage Q des règles de transformation pour les transformations *prog662AsBuilt* et *TableDE2AsBuilt*

Comme nous pouvons le constater, les transformations *prog662AsBuilt* et *TableDE2AsBuilt* ont parfaitement été apprises. Une phase de validation des règles de transformation a été réalisée en réutilisant les matrices de transformation inférées pendant la phase d'apprentissage pour de toutes nouvelles instances de modèles (pour de nouveaux équipements différents des Gyro). Les temps d'apprentissage sont, encore une fois, très faibles, ce qui permet un déploiement rapide de la solution d'automatisation proposée.

Détaillons plus précisément les résultats obtenus de la transformation *prog662AsBuilt*.

La principale difficulté de la transformation *prog662AsBuilt* consiste à exprimer de manière significative la relation hiérarchique entre la ligne n°1 et la ligne n°3 du fichier *prog66* (voir ligne en rouge dans la Figure 5.16), à savoir, la relation entre un article dit "fils" et un article dit "père". Cette structure hiérarchique est exprimée sous la forme d'une nomenclature produit, où, l'article père GYRO EQUIPE est un sous-ensemble du satellite, tandis que l'article fils GYRO XXX est l'un des composants qui permet d'assembler l'article père GYRO EQUIPE (d'où la notion de Gyro dit "équipé"). L'agent en charge d'apprendre les règles de transformation entre les modèles *prog66* et *AsBuilt* doit être en capacité de résoudre deux conditions de transformation de type σ_r, à savoir : (1) Seuls les articles fils sont renseignés dans le document *As-Built* (éléments surlignés en jaune dans la Figure 5.16) ; (2) La colonne "Topo n+1" du document *As-Built* correspond au repère topologique du satellite où est

intégré l'article père (élément surligné en bleu dans la Figure 5.16).

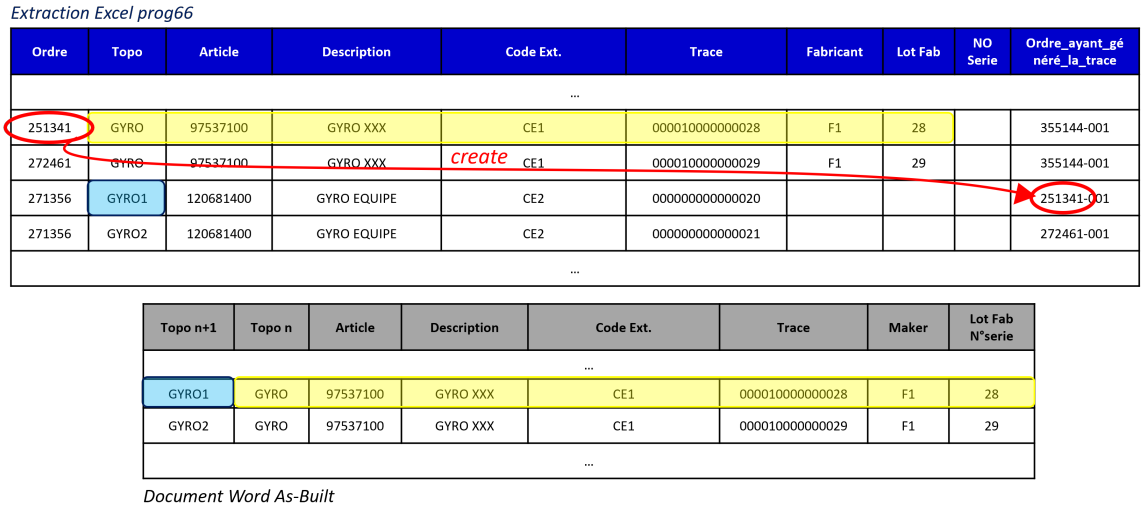


FIGURE 5.16 – Condition de transformation σ_r de la transformation *prog662AsBuilt*

Pour modéliser la relation hiérarchique entre l'article père GYRO EQUIPE et l'article fils GYRO XXX, selon le métamodèle *prog66.ecore* de la Figure B.3, une référence de type *create* relie les ordres de fabrication (OF) de l'article père et de l'article fils. De ce fait l'OF 251341 contenant l'article GYRO XXX permet la création de l'OF 271356 contenant l'article GYRO EQUIPE.

Pour rappel, la première étape du processus d'apprentissage des modèles de transformation consiste à partitionner les modèles *prog66* et *AsBuilt* de sorte à regrouper les classes qui ont des contextes interne et externe identiques. Les Figures 5.18 et 5.17 présentent les classes aplaties obtenues après le processus de partitionnement des modèles.

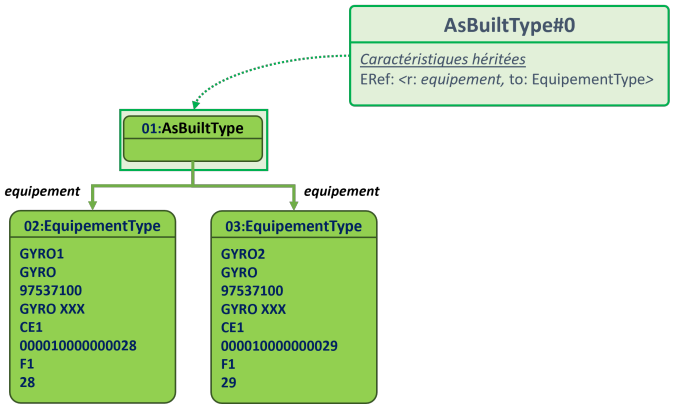


FIGURE 5.17 – Classes aplaties du métamodèle *AsBuilt*

Une fois de plus, le partitionnement des modèles sous forme de représentations aplaties des classes semble être suffisant pour exprimer les spécificités de la structure des modèles. Comme nous pouvons

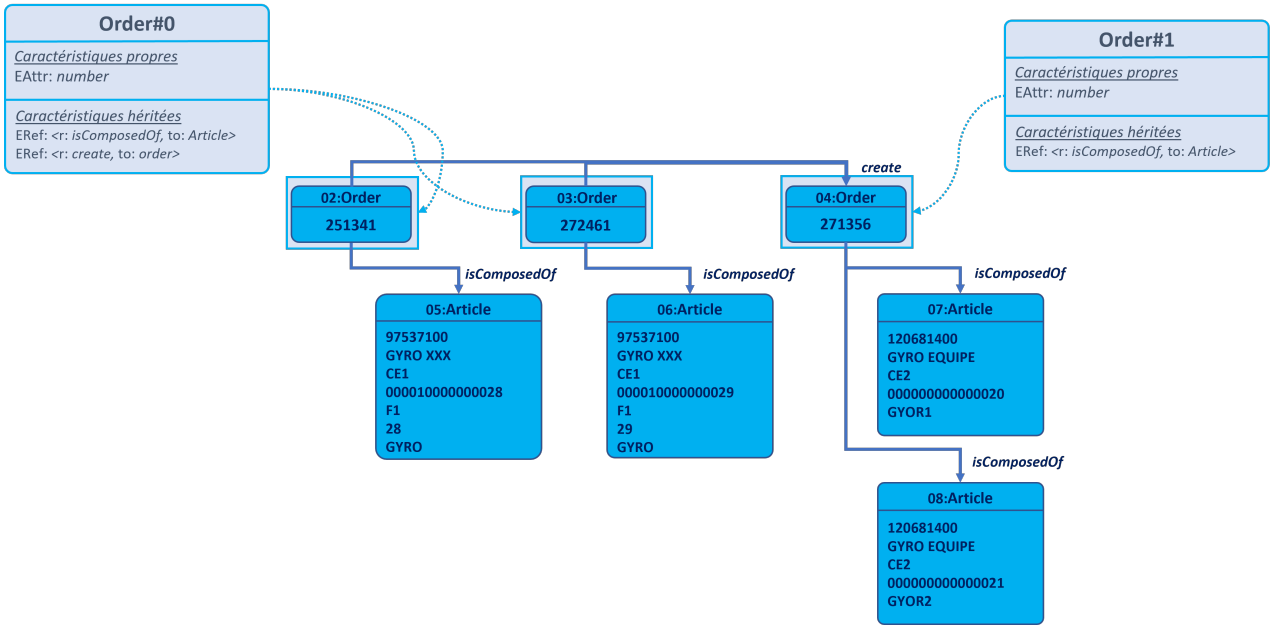


FIGURE 5.18 – Classes aplaties du métamodèle *prog66*

le constater, le modèle *prog66* est divisé selon deux classes aplaties. La classe aplatie Order#0 recense les classes de type [*Order*] ,qui disposent d'une référence particulière de type *create*, avec d'autres classes de type [*Order*] ; tandis que la classe aplatie Order#1 recense les classes de type [*Order*] qui ont été créées. Concernant les classes de type [*Article*], aucune classe aplatie n'est générée car ces dernières ne comportent pas de contexte à droite, à savoir, de références avec d'autres classes.

La matrice de transformation de la Figure 5.19, générée par l'agent une fois l'apprentissage terminé, permet d'analyser les règles de transformation inférées. Nous pouvons ainsi constater que seuls les éléments propres au contexte de la classe aplatie Order#0 sont créés, à savoir, les éléments relatifs aux articles fils GYRO XXX. L'ensemble des éléments propres à la classe aplatie Order#1 sont reliés au pattern *None*, ce qui signifie que pour un pattern source, aucune équivalence n'existe dans le domaine du métamodèle cible. L'unique exception concerne le pattern source $[1, Article, topo]$ qui correspond à l'information du repère topologique sur lequel le sous-ensemble GYRO EQUIPE est monté. Nous pouvons constater que ce dernier est bien relié au pattern cible $[0, EquipementType, TopoN1]$ où l'attribut de type *TopoN1* correspond à la colonne "Topo n+1" du tableau du document *As-Built*. Par conséquent, l'agent à bel et bien réussi à résoudre les patterns de transformation soumis à une condition en relation de type σ_r.

Nous pouvons alors conclure que l'automatisation des transformations *prog662AsBuilt* et *TableDE2AsBuilt* a permis de réduire considérablement le temps nécessaire pour éditer un document de type *As-Built*. Le processus de transformation facilite la traduction sémantique des messages échangés entre un service producteur de données et le service qui va consommer ces données.

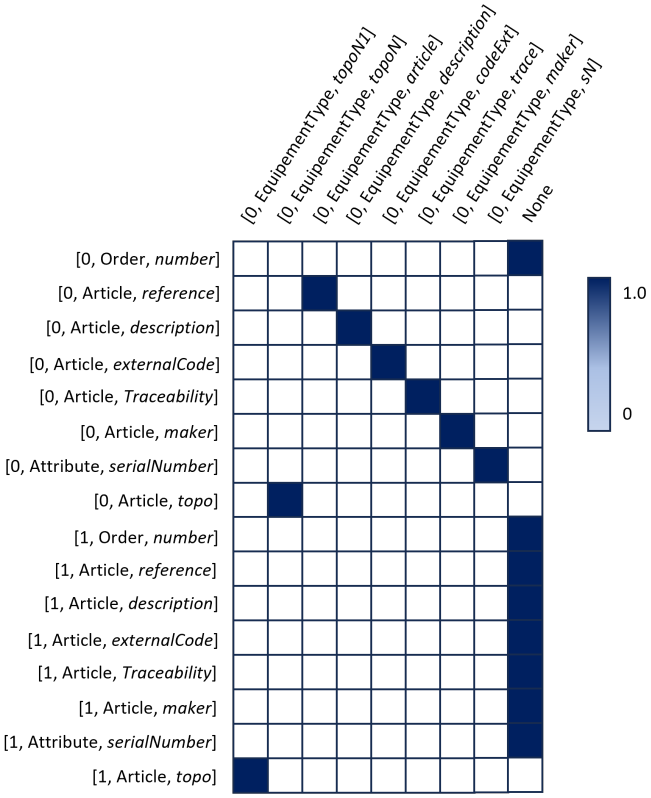


FIGURE 5.19 – Matrice de transformation pour la transformation *prog662AsBuilt*

5.2.4 Cas d'étude n°2 : Automatisation de la planification des activités pendant les phases d'AIT

5.2.4.1 Description du contexte

Contexte. L'Industrie 4.0, et plus particulièrement l'industrie intelligente dirigée par les données, introduit de nouvelles perspectives pour les entreprises du secteur manufacturier. L'émergence de nouvelles technologies, qui découlent des concepts clés de l'Industrie 4.0, procure de multiples opportunités, notamment concernant les activités de planification, de supervision et de contrôle de la production [31]. Généralement, la notion de planification est utilisée lorsqu'il s'agit d'allouer des ressources à une activité pendant une période prédéfinie pour réaliser un objectif. La notion de supervision, quant à elle, permet, par des moyens visuels, d'observer et d'analyser en temps réel l'encours d'une activité. Enfin, la notion de contrôle fait référence à la nécessité d'agir sur une activité pour en contrôler son évolution. *In fine*, la planification, la supervision et le contrôle de la production permettent de réduire les coûts et les ressources engagées, les risques, d'améliorer l'efficacité et de gérer la complexité de toute une chaîne de production.

L'objectif d'un système en charge de la planification des activités est de s'auto-ajuster automatiquement et en toute autonomie, et ce, selon les évolutions, les changements et les modifications de

l'environnement dans lequel il intervient. Ces dernières propriétés mettent en évidence la nécessité d'un système de planification à correctement appréhender les environnements dynamiques, pour résoudre des objectifs faisant intervenir différents métiers, différentes organisations et différents systèmes (physiques et numériques).

Le concept d'industrie intelligente repose inévitablement sur des systèmes intelligents capables de planifier, de superviser et de contrôler les activités de la production [197]

C'est en ce sens que Thales Alenia Space a entamé différents chantiers pour améliorer la qualité, la fiabilité et l'automatisation de la planification des activités pendant les phases d'AIT. Les phases d'AIT s'intègrent dans un environnement décisionnel complexe qui fait intervenir de multiples métiers, différents systèmes d'information, ainsi que plusieurs satellites en parallèle en cours d'assemblage, d'intégration et de test. Par souci de simplicité, le cas d'étude présenté dans cette section sera appliqué à la planification des activités mécaniques (intégration d'instruments, assemblage de panneaux...) pour un unique satellite.

La Figure 5.20 présente l'ensemble des systèmes de la production qui interviennent lors du processus de planification des activités mécaniques pendant les phases d'AIT.

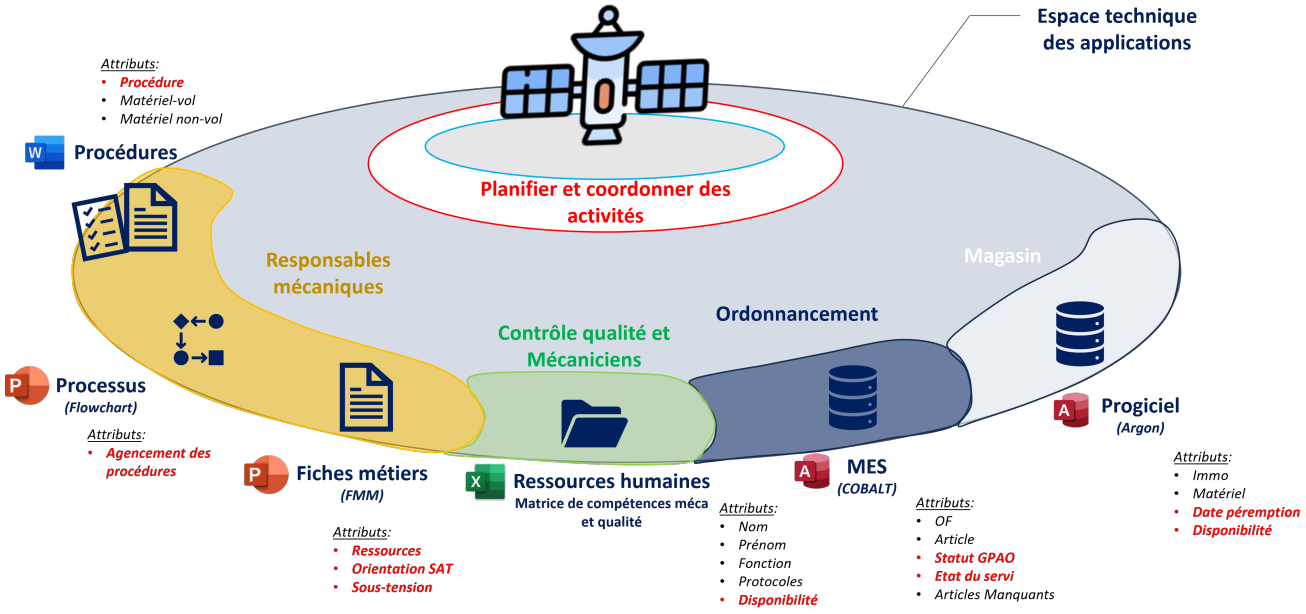


FIGURE 5.20 – Vue d'ensemble des systèmes intervenant dans le processus de planification des activités mécaniques pendant les phases d'AIT

Globalement, planifier des activités d'assemblages et d'intégrations dans un contexte d'AIT consiste à ordonnancer une série d'activités, ou procédures, à réaliser en fonction de la disponibilité des ressources. Le chemin critique de cette succession de procédures est prédéfini par un diagramme de séquence. Les ressources nécessaires pour démarrer une procédure sont de trois natures :

- *Humaines :* L'intégralité des ressources humaines pour le métier de la mécanique est recensée dans des matrices de compétences au format *Excel*, image (PNG) et PDF. Ces matrices désignent les compétences détenues par chaque mécanicien. Par exemple, selon la procédure à réaliser, un protocole particulier peut être nécessaire. Le choix de la ressource humaine pour réaliser l'activité s'orientera alors vers un mécanicien qui détient la certification.
- *Matériel vol :* Le matériel vol constitue un ensemble de matériel à assembler ou à intégrer sur le satellite. Il s'agit généralement de sous-systèmes (antennes, panneaux, *star tracker*...) ou de visseries. La liste des articles nécessaires est répertoriée dans le système MES COBALT qui gère la gestion des ordres de fabrication.
- *Matériel non-vol :* Les matériels non-vol de type MGSE (*Ground Support Equipment*) et EGSE (*Electrical Ground Support Equipment*) sont des équipement utilisés au sol pour soutenir les opérations d'assemblage, d'intégration et de tests des satellites. Les équipements MGSE incluent les simulateurs, les systèmes d'alimentation électrique, les bancs de tests et tous dispositifs de manutention. Tandis que les équipements EGSE sont des dispositifs nécessaires pour vérifier que tous les systèmes électriques fonctionnent correctement. L'ensemble de ces équipements est référencé dans le progiciel ARGON de gestion des équipements non-vol.

Problème. Un système de planification automatique nécessite obligatoirement d'être connecté aux systèmes de la production pour acquérir en temps réel les données de l'encours des activités, à savoir, les caractéristiques de l'environnement à contrôler [144]. L'acquisition des données provenant de différentes sources d'informations est inévitable pour assurer une prise de décision éclairée, qui repose sur les données [111]. Or, assurer l'interopérabilité entre une multitude de systèmes hétérogènes est une activité complexe.

Actuellement, la récupération des différentes données est réalisée à la main par le métier de responsable mécanique. Il est en charge d'agréger les données pour obtenir une vue d'ensemble de l'encours des activités. Pour réunir l'intégralité des données, le responsable mécanique doit jongler entre les différents systèmes d'information. Cependant, l'environnement de l'AIT est extrêmement dynamique. Cela signifie que les données acquises ne sont valables qu'à l'instant t où elles ont été récupérées. La volatilité des données est un phénomène important à considérer lorsqu'il s'agit d'établir une planification des activités en temps réel, et d'apporter une réponse rapide aux changements. Par conséquent, une planification manuelle qui repose sur l'acquisition des données par un humain est inévitablement obsolète au premier changement de l'environnement.

Proposition. Afin d'apporter une vision détaillée et en temps réel de la disponibilité des ressources nécessaires pour démarrer une procédure, nous proposons d'automatiser le processus qui consiste à récupérer, pour une liste de matériels vols et non-vols spécifiée par une procédure, leurs statuts de disponibilité. Une fois de plus, une approche orientée services exploitant les principes de l'architecture dirigée par les modèles, telle que présentée en Section 5.1.2, sera à nouveau utilisée. Pour répondre aux problématiques de dynamicité de l'environnement industriel, où la disponibilité des ressources varie en fonction de la demande, une architecture dirigée par les événements sera adoptée. La Figure 5.21 présente une vue d'ensemble de la proposition. Une architecture orientée événements est composée de systèmes qui interagissent en réagissant à des événements. Dans le cas présent, les systèmes im-

pliqués sont le MES COBALT, le progiciel ARGON ainsi que les procédures mécaniques spécifiant la séquence d'action à réaliser pour mener à bien une activité. Les évènements qui permettront d'initier la communication entre deux services prendront deux formes :

- *Une Action utilisateur :* Une action utilisateur est rendue possible via une interface graphique (voir main orange dans la Figure 5.21). L'utilisateur pourra sélectionner une procédure et visualiser en temps réel la disponibilité du matériel associé. La récupération des informations dans les différents systèmes suivra un processus en deux temps : (1) Pour commencer, selon le numéro de procédure sélectionné, un premier processus sera en charge de récupérer la liste des matériels vols et non-vols ; (2) Une fois la liste des matériels vols et non-vols récupérée, un second processus sera attitré à la récupération des statuts de disponibilité des matériels dans les systèmes COBALT et ARGON.
- *Un changement d'état :* Dans le cas où le statut de disponibilité d'un matériel venait à changer, l'interface graphique utilisateur sera mise à jour automatiquement (voir clochette orange dans la Figure 5.21).

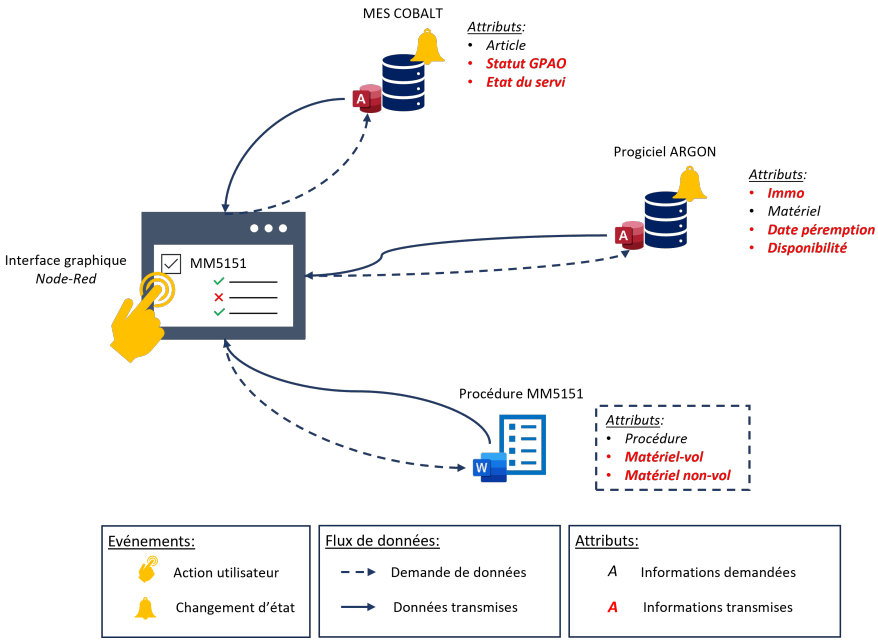


FIGURE 5.21 – Vue d'ensemble de la proposition

Une architecture orientée évènements permet le développement d'application temps réel où les systèmes fonctionnent de manière asynchrone. L'outil de développement *Node-red*[1] est utilisé pour construire une interface utilisateur graphique ergonomique permettant la visualisation des données issues du MES COBALT et du progiciel ARGON. *Node-red* permet d'automatiser la communication entre les différents services et d'assurer un flux de données continu. En d'autres termes, la continuité numérique sera assurée par l'intermédiaire d'une architecture logicielle qui orchestre la communication

1. `https://nodered.org/`

entre différents services, où la communication entre les services est régie par des événements.

5.2.4.2 Description technique de la solution

La solution technique permettant d'automatiser la récupération des données des matériels vols et non-vols associés à une procédure suivra les spécifications de l'architecture logicielle d'une application présentée en Section 5.1.2. L'objectif de l'application est : (1) D'automatiser la récupération de la liste des matériels vols et non-vols d'une procédure mécanique ; (2) D'automatiser la récupération des statuts de disponibilité des matériels associés à la procédure ; (3) De centraliser les données issues de plusieurs systèmes au sein d'une même interface graphique. Pour cela, l'application sera composée de quatre services :

- *Le service producteur COBALT* sera en charge de récupérer l'état de servi d'une liste d'article (matériels vols). En gestion de production, l'état de servi magasin indique si le matériel a été sorti des stocks et mis à disposition du programme (du satellite) qui en fait la demande. Un état de servi "complet" indique que le matériel a été servi dans son intégralité et est disponible au magasin pour être retiré. La méthode *get_EtatServi()*, prenant en paramètre une liste d'articles, sera utilisée pour interroger la base de données *Access* du MES COBALT, puis pour récupérer les informations concernant les états de servi des articles. Évidemment, un mécanisme d'injection propre à la plateforme *Access* a été développé pour permettre au service *COBALT* d'interagir avec la base de données du MES COBALT.
- *Le service producteur ARGON* sera, quant à lui, en charge de fournir les informations de disponibilité et de péremption relatives à un matériel non-vol. Un matériel de type EGSE et MGSE peut être attitré à un unique programme. Le statut de disponibilité indique donc si le matériel est disponible au magasin, ou s'il est déjà empreinté par un autre programme. Par ailleurs, certains matériels non-vols sont soumis à des dates de péremption. Pour être empreinté, la date de péremption du matériel ne doit pas être dépassée. La méthode *get_availability()*, prenant en paramètre une liste de matériels non-vols, sera utilisé pour interroger la base de données *Access* du progiciel ARGON, puis pour récupérer les informations concernant la disponibilité des matériels non-vols.
- *Le service producteur ProcMECA* (pour procédure mécanique), donnera un accès au contenu des procédures mécaniques au format *Word*, i.e. un accès aux listes des matériels vols et non-vols. La méthode *get_materials()*, prenant en paramètre un numéro de procédure, sera utilisée pour interroger le fichier *Word* propre à la procédure mécanique, puis pour récupérer la liste des matériels vols et non-vols. Un mécanisme d'injection propre à la plateforme *Word* a été développé pour permettre au service *ProcMECA* d'interagir avec les fichiers *Word* des procédures.
- *Le service consommateur ProcVisualisation* sera en charge de récupérer les données issues des services *COBALT*, *ARGON* et *ProcMECA* pour les afficher au sein d'une interface graphique permettant leur visualisation.

Intéressons nous plus particulièrement au service consommateur *ProcVisualisation* en charge d'agréer les données issues des services *COBALT*, *ARGON* et *ProcMECA* et de les afficher au sein d'une interface utilisateur graphique. Le diagramme de classe du service *ProcVisualisation* est présenté dans

la Figure 5.22.

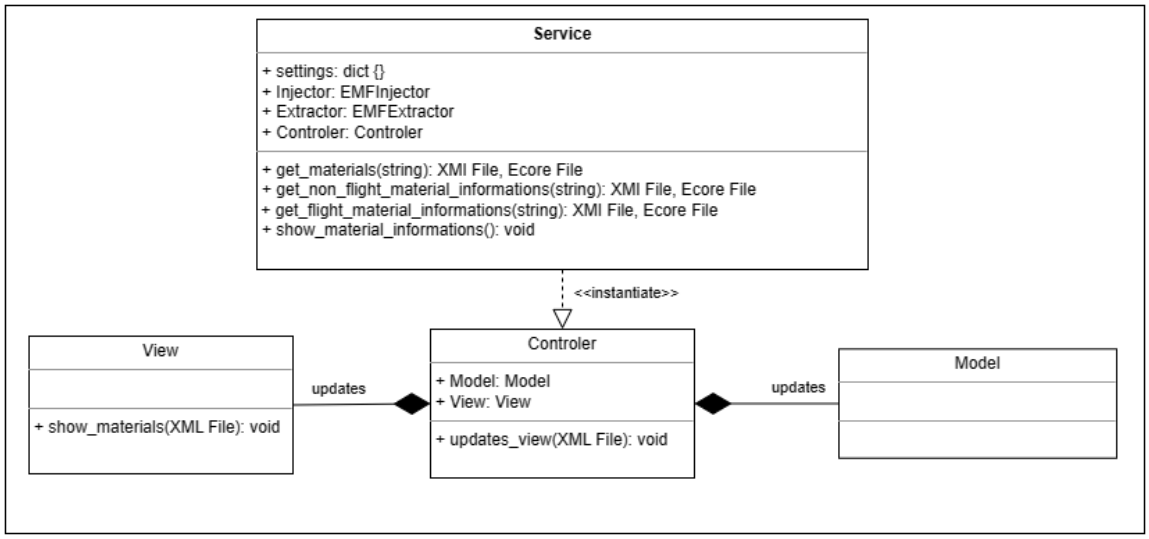


FIGURE 5.22 – Diagramme de classes du service *ProcVisualisation*

Détaillons le diagramme de classe du service *ProcVisualisation* :

- La méthode *get_materials()* permet d'interroger le service *ProcMECA* pour récupérer la liste des matériels vols et non-vols d'une procédure.
- La méthode *get_non_flight_material_informations()* permet d'interroger le service *ARGON* pour récupérer le statut de disponibilité du matériel non-vol de la procédure.
- La méthode *get_flight_material_informations()* permet d'interroger le service *COBALT* pour récupérer l'état de servi du matériel vol de la procédure.

Étant donné que l'utilisateur sélectionne un numéro de procédure depuis l'interface graphique de *Node-Red*, l'évènement "action utilisateur" déclenche la méthode *show_material()* de la classe *View*, où, le numéro de procédure sélectionné par l'utilisateur est transmis au format XML au service *ProcVisualisation*. Les trois méthodes précédentes du service *ProcVisualisation* sont alors déclenchées par la classe *Controler* en charge de fournir les informations nécessaires pour mettre à jour la vue.

Pour assurer la communication entre le service *ProcVisualisation* et les services *ProcMECA*, *ARGON* et *COBALT*, deux mécanismes de connexion doivent être appris. Le premier consiste à établir la connexion entre le service *ProcVisualisation* et *ProcMECA* lors de l'appel de la méthode *get_materials()*. Le second consiste à établir la connexion entre le service *ProcVisualisation* et les services *ARGON* et *COBALT* lors de l'appel des méthodes permettant la récupération des informations concernant les matériels vols et non vols. Étant donné que le métamodèle du service *ProcVisualisation* est un métamodèle qui agrège à la fois les données issues du métamodèle du service *ARGON* et du métamodèle du service *COBALT*, une unique transformation peut être apprise.

Les Figures 5.23 et 5.24 illustrent respectivement les transformations *ProcMECA2ProcVisualisation* et *COBAT_ARGON2ProcVisualisation*.

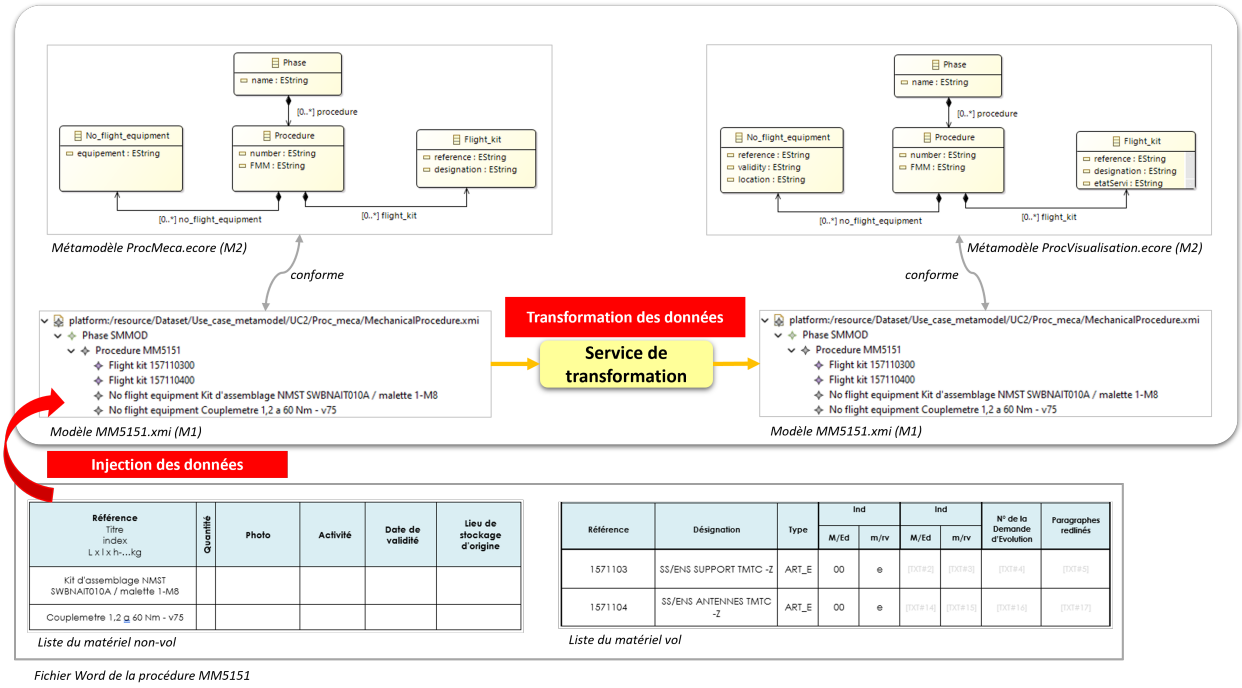


FIGURE 5.23 – Vue d'ensemble de la transformation *ProcMECA2ProcVisualisation*

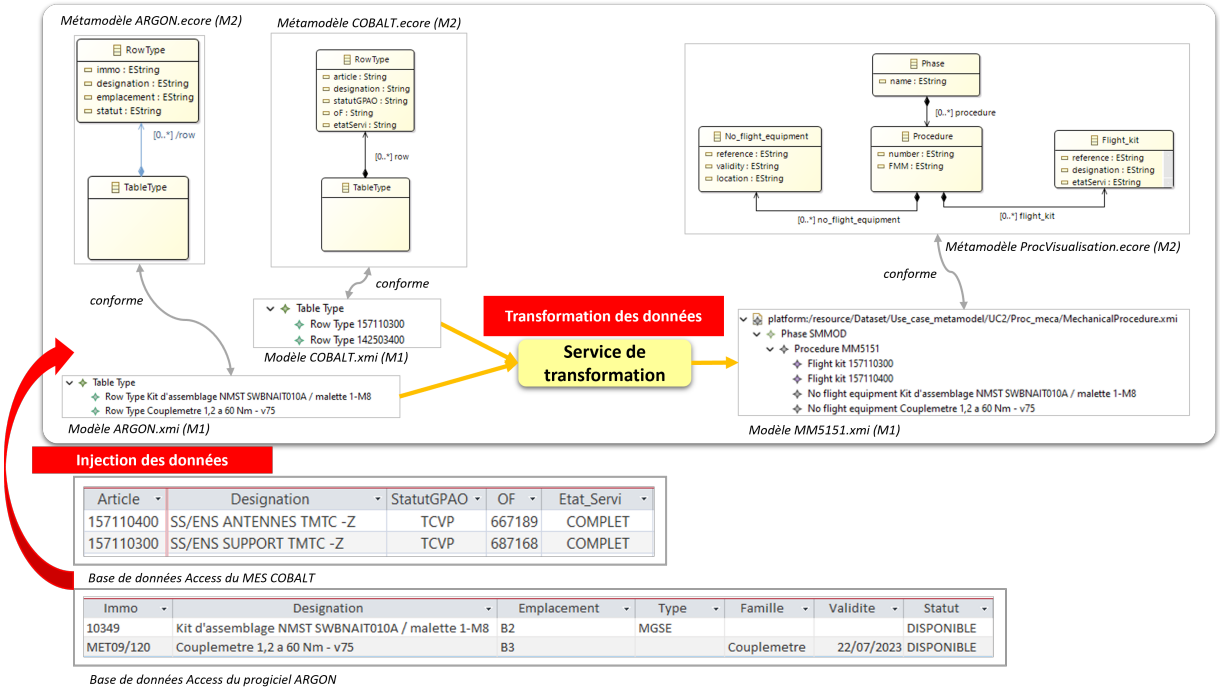


FIGURE 5.24 – Vue d'ensemble de la transformation *COBAT_ARGON2ProcVisualisation*

Pour simplifier le processus d'apprentissage des deux modèles de transformation, les bases de

données du MES COBALT et du progiciel ARGON ont été filtrées sur des articles vols et des matériels non-vols appelés par la procédure MM5151. Le métamodèle d'agrégation *ProcVisualisation.ecore* ayant été construit manuellement, le modèle *MM5151.xmi* a également été instancié à la main. Étant donné que l'approche d'apprentissage des modèles de transformation s'appuie sur des exemples de modèles source et cible, lors de la création d'un tout nouveau métamodèle (dans le cas présent le métamodèle *ProcVisualisation.ecore*), il est cohérent de construire manuellement l'instance du modèle cible souhaité après transformation. Il est souvent plus rapide de construire le modèle cible souhaité plutôt que d'éditer des règles de transformation pour l'obtenir. C'est sur ce paradigme que repose toute l'approche d'apprentissage des modèles de transformation.

Le Tableau 5.3 présente les patterns de transformation qui interviennent lors des transformations de modèles *ProcMECA2ProcVisualisation* et *COBAT_ARGON2ProcVisualisation*. Comme nous pouvons le constater, des patterns de transformation très basiques de type (c1) interviennent.

	Caractéristiques structurelles							***Caractéristiques terminologiques***			
	Patterns de transformation				*Conditions*						
Transformation	$[1-1]$	$[1-n]$	$[n-1]$	$[n-m]$	σ_c	σ_a	σ_r	C	R	A	V
Families2Persons	-	-	(c2)	-	-	-	x	(e4)	(e4)	(e3)	-
Class2Relational	(c1)	(c5)	-	(c4)	x	x	-	(e6)	(e6)	(e6)	(d2 ;d5)
ProcMECA2ProcVisu	(c1)	-	-	-	-	-	-	-	-	-	-
CO_AR2ProcVisu	(c1)	-	-	-	-	-	-	(e6)	(e6)	(e6)	-

TABLE 5.3 – Caractérisation des transformations de modèles du cas d'étude n°2

Une fois le modèle *MM5151.xmi* obtenu après la transformation *COBAT_ARGON2ProcVisualisation*, l'extracteur spécifique à EMF permet d'obtenir le fichier *MM5151.xml* au format XML. Ce dernier est ensuite transmis, par l'intermédiaire d'un serveur MQTT, à la plateforme *Node-Red* qui sera en charge de mettre en forme les données de la procédure MM5151 dans l'interface graphique. La Figure 5.25 présente le principe de fonctionnement d'un protocole de communication MQTT. Le service *ProcVisualisation* publie le modèle de données *MM5151.xml* issu de la transformation *COBAT_ARGON2ProcVisualisation* sur un serveur MQTT auquel la plateforme *Node-Red* va souscrire pour récupérer les informations de la procédure.

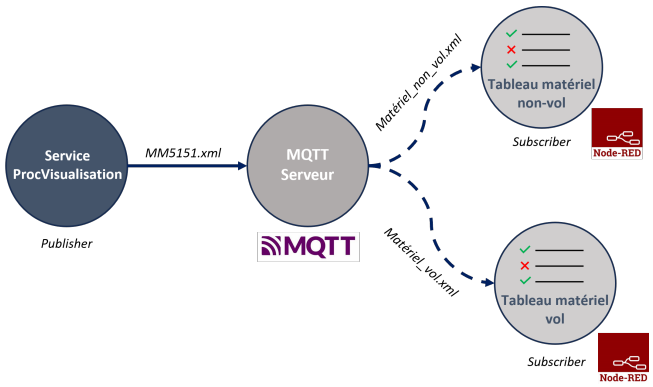


FIGURE 5.25 – Principes d'un protocole de communique MQTT

5.2.4.3 Résultats

Les résultats des transformations *ProcMECA2ProcVisualisation* et *COBAT_ARGON2ProcVisualisation* sont présentés dans le Tableau 5.4. Sans surprise, les règles de transformation ont correctement été apprises par l'agent. Il est tout à fait attendu que les modèles de transformation aient été déduits dans leur intégralité, étant donné que seuls des patterns de transformation de type (c1) interviennent pendant les deux transformations. Par conséquent, la communication entre les différents services a pu être établie. La continuité numérique entre le MES COBALT, le progiciel ARGON et les procédures mécaniques, favorise une planification en temps réel des activités d'assemblages et d'intégrations mécaniques, en donnant accès à l'utilisateur à des données constamment actualisées.

	Performances d'apprentissage				***Performance de transformation***
Transformation	*Temps (s)*	*P*	*R*	*F*	*F*
ProcMECA2ProcVisualisation	0.46	1.0	1.0	1.0	1.0
COBAT_ARGON2ProcVisualisation	1.01	1.0	1.0	1.0	1.0

TABLE 5.4 – Résultats de l'apprentissage Q des règles de transformation pour les transformations *ProcMECA2ProcVisualisation* et *COBAT_ARGON2ProcVisualisation*

L'interface graphique utilisateur permettant la visualisation des informations concernant la liste des matériels vols et non-vols d'une procédure est présentée dans la Figure 5.26. Tout à gauche, l'utilisateur sélectionne la ou les procédures qu'il veut visualiser. L'événement 'action utilisateur' est ensuite déclenché, ce qui permet d'activer les services responsables de la récupération des données concernant la ou les procédures choisies. Deux objets graphiques de type tableau, sont utilisés pour afficher le résultat de la récupération des données. Le premier tableau, *Kitting informations*, donne des indications sur l'état de servi du matériel vol, tandis que le second tableau, *MGSE informations*, donne des indications sur la disponibilité du matériel non-vol. A travers cette interface graphique, l'utilisateur a un accès centralisé à l'ensemble des ressources matériels nécessaires au démarrage d'une procédure. L'interface graphique permet ainsi, de visualiser des données provenant de différentes sources d'informations.

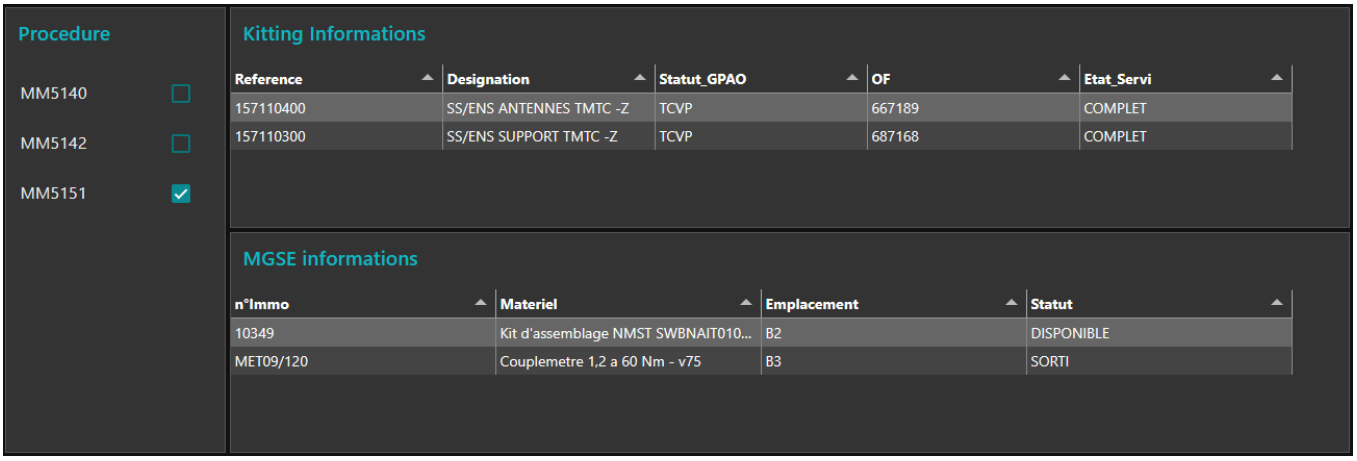


FIGURE 5.26 – Interface graphique utilisateur depuis la plateforme *Node-Red*

Chapitre 6

Conclusions et perspectives

Contenu

6.1 Conclusion

Environnement d'étude. Promouvoir la continuité numérique dans un contexte d'Industrie 4.0 nécessite de répondre à des contraintes (1) techniques causées par la démultiplication et l'hétérogénéité des supports de l'information ; (2) organisationnelles créées par une demande accrue de réactivité, d'efficacité et de flexibilité dans la reconfiguration de la chaîne de valeur d'une entreprise ; (3) économiques engendrées par une forte demande de personnalisation des services et par la nécessité de travailler avec de multiples acteurs industriels répartis à travers le monde. L'interopérabilité des systèmes est un enjeu majeur pour soutenir la continuité numérique de l'information et permettre l'émergence de l'industrie intelligente dirigée par les données, à travers la coopération des systèmes physiques et numériques de la production.

Besoins de l'environnement. L'étude des grandes approches pour l'interopérabilité (approches intégrée, unifiée et fédérée), a démontré que l'approche fédérée est la solution qui répond le mieux aux spécifications de l'Industrie 4.0, en favorisant l'élaboration de Systèmes Cyber-physiques, ou, plus communément appelé Système de systèmes. Une architecture fédérée garantit un couplage lâche entre les systèmes et préserve les propriétés d'autonomie, d'évolutivité, de dynamicité et de diversité des systèmes qui participent à la fédération. Une fédération est une architecture modulaire qui facilite la suppression, la modification et l'ajout de nouvelles sources d'informations. Une interopérabilité *plug and play* est donc nécessaire pour permettre un couplage dynamique, à la volée, entre les systèmes. A partir de l'analyse des caractéristiques et des concepts portés par l'Industrie 4.0, les grands défis de l'interopérabilité ont été définis, à savoir, la création d'un mécanisme de connexion générique (indépendant de toutes solutions techniques), automatique (création d'une fonction d'interopérabilité à la volée) et dynamique (capable d'évoluer en fonction des changements apportés à son environnement) de sorte à garantir l'interopérabilité *plug and play* tant souhaitée.

Proposition fondamentale. Afin d'apporter une solution concrète aux défis identifiés, la proposition portée par ce manuscrit s'inscrit dans une démarche où l'interopérabilité est dite dirigée par les modèles. Le concept d'interopérabilité présenté suit les principes énoncés par l'Architecture Dirigée par les Modèles (MDA) qui spécifie que, tout système, est représenté par un ou plusieurs métamodèles (M2), qui expriment la structure et la sémantique des modèles de données (M1). Les modèles de transformation constituent la pierre angulaire de l'approche MDA et permettent la transformation des métamodèles d'un domaine vers un autre, et dans une moindre mesure, garantissent l'interopérabilité entre les modèles de données. Les transformations "modèles vers modèles" jouent alors le rôle de fonction d'interopérabilité entre les modèles, et par extension, entre les systèmes. L'interopérabilité entre les systèmes est alors rendue possible par l'intermédiaire de l'espace technique des modèles.

Initialement décrites à la main, les règles de transformation qui permettent le passage d'un métamodèle vers un autre ne répondent pas aux besoins spécifiés par l'Industrie 4.0. Une proposition fondamentale a donc été soumise, selon laquelle, les récentes avancées dans le domaine de l'intelligence artificielle pourraient simplifier ou automatiser la construction d'un modèle de transformation pour satisfaire les attentes d'une interopérabilité générique, automatique et dynamique. Deux hypothèses fortes ont donc été posées : (1) Le mécanisme de connexion permet une interopérabilité automa-

tique entre les systèmes, par l'intermédiaire de leurs métamodèles, et ce, en minimisant les pertes sémantiques ; (2) Le mécanisme de connexion satisfait les spécifications d'une interopérabilité *plug and play* en minimisant les efforts humains pour créer le connecteur en termes de pré-traitement et post-traitement, et en permettant toutes évolutions et modifications de son environnement.

Une approche d'apprentissage des modèles de transformation, à savoir des règles de transformation, a été expérimentée pour inférer les relations structurelles et sémantiques qui relient deux métamodèles. L'approche d'apprentissage s'appuie sur les techniques de l'apprentissage par renforcement, et plus particulièrement sur les principes de l'apprentissage Q.

Validation académique. Une partie de la proposition de ce manuscrit visait à construire un protocole expérimental permettant d'évaluer les différentes approches d'apprentissage des modèles de transformation, vis-à-vis de critères de performance objectifs, en phase avec les hypothèses précédemment dressées. Le protocole proposé contient des jeux de données caractérisés, des critères de validation, ainsi qu'une grille d'évaluation permettant de caractériser la complexité des transformations à mettre en œuvre du point de vue des conflits structuraux et terminologiques qui peuvent intervenir lors d'une transformation de modèle.

Suivant le protocole expérimental, une étude de type *benchmark* a été réalisée et a montré que les résultats obtenus étaient prometteurs, surclassant les performances des approches d'apprentissage des modèles de transformation existantes jusqu'à présent. L'approche proposée a été évaluée selon des critères de performances et des cas d'études prédéfinis. L'apprentissage par renforcement semble être une solution viable pour assurer une interopérabilité générique et automatique entre les métamodèles, et par extension, entre les systèmes. Le modèle de transformation généré est indépendant de tout langage de transformation spécifique et de tout métamodèle à transformer (propriété de généricité). Il est également généré de manière automatique avec, pour les jeux de données de validation exploités, une perte sémantique nulle, et des efforts humains de mise en oeuvre très faibles (propriété d'automatisation). En revanche, la propriété de dynamisme n'est pas directement résolue dans l'étude proposée. Dans le cas où un connecteur doit absorber les changements et les évolutions de l'un des systèmes, la solution actuelle propose de ré-exécuter la phase d'apprentissage pour intégrer les nouvelles règles de transformation.

Validation industrielle. L'approche d'apprentissage par renforcement des modèles de transformation, pour assurer l'interopérabilité sémantique entre deux systèmes, a été implémentée dans une architecture orientée services. Un service de transformation encapsule donc l'ensemble de la proposition de ce manuscrit. Une architecture orientée services, dirigée par les modèles, a donc été proposée pour permettre l'utilisation du service de transformation grâce à des services d'injection et d'extraction permettant le passage de l'espace technique des systèmes, vers l'espace technique des modèles, et inversement.

Deux preuves de concepts ont été construites à partir de cas d'étude provenant de l'industrie spatiale. L'une porte sur l'automatisation de documents de type *As-Built*, l'autre porte sur la planification intelligente des activités d'assemblage, d'intégration et de tests des satellites. Toutes deux ont prouvées la pertinence de la proposition décrite dans ce manuscrit en automatisant des processus autre fois complexes à réaliser par l'humain, tout en garantissant des résultats. Ces cas d'études ont également

permis de mettre en lumière les manques et les verrous à résoudre pour pouvoir déployer à plus large échelle le transformateur proposé.

6.2 Perspectives

Plusieurs perspectives sont envisageables pour enrichir la solution proposée dans ce manuscrit.

Méthodes de transformation de graphes. Vis-à-vis des limitations identifiées en Section 4.5, il semble clair que l'architecture proposée nécessite des améliorations concernant la méthode permettant de capturer la représentation des classes. En effet, une représentation fidèle et explicite des classes permet de simplifier le processus d'alignement et, *in fine*, d'inférer une règle de transformation complète.

A l'image du problème de traduction automatique du langage, tant étudié par la communauté du traitement naturel du langage (*Natural language processing* NLP), où la représentation vectorielle d'un mot d'une phrase encapsule à la fois son contexte à droite et à gauche (i.e, l'ensemble des mots qui l'entourent) [11], la représentation vectorielle d'une classe devrait également considérer l'ensemble des classes à droite et à gauche qui l'entourent, et ce, quelque soit la profondeur. Dans leur récente étude, Guo *et al.* [69] proposent un nouveau domaine d'étude nommé "la transformation de graphe" qui s'appuie principalement sur deux méthodes :

- *Approches basées sur l'apprentissage de représentation.* Le graphe source est encodé dans un espace vectoriel latent à l'aide d'un encodeur, où, l'intégralité du patrimoine informationnel de chaque entité est représentée sous la forme d'un vecteur. Le vecteur est ensuite décodé à l'aide d'un décodeur afin d'obtenir le graphe cible souhaité.
- *Approches basées sur l'édition d'un graphe.* Le graphe cible est édité de façon itérative.

Ces méthodes exploitent la puissance des GNN pour leur capacité à créer un vecteur de représentation fidèle pour chacune des entités d'un graphe de données.

Validation. Afin d'enrichir la grille de caractérisation des transformations de modèles avec de nouveaux patterns de transformations, des jeux de données ont été identifiés : (1) La transformation d'un diagramme de séquence UML vers un réseau de pétrit coloré [26] ; (2) La transformation d'un diagramme de séquence UML vers un diagramme d'état [68]. Le diagramme de séquence UML comporte des *fragments d'interaction* qui modélisent un comportement complexe. Les fragments d'interaction sont des représentations permettant de modéliser un comportement alternatif, itératif et parallèle d'une suite d'actions. Concernant le diagramme en réseau de pétrit coloré, il met en évidence, à l'aide de couleurs, les différents type d'objets qui interviennent. Enfin, un diagramme d'état est une représentation d'un processus qui se produit lors du fonctionnement d'une machine. L'utilité principale de ce diagramme est de visualiser les performances d'un objet lorsqu'il subit une opération. Cela signifie qu'il montre des données précieuses sur la réaction de la machine, aux différents états qu'elle subit à chaque opération.

Benchmark. Intégrer le *Benchmark* de l'*Ontology Alignment Evaluation Initiative* (OAEI) dans la section propre aux solutions basées sur l'alignement des instances de modèle pourrait permettre de mettre en évidence les bienfaits et les lacunes de la solution proposée dans ce manuscrit. L'OAEI propose des jeux de données ainsi que des indicateurs de performance permettant d'évaluer si le

mappage inféré est conforme au mappage attendu.

Bibliographie

[1] R. Abbott. "Open at the Top ; Open at the Bottom ; and Continually (but Slowly) Evolving". en. In : *2006 IEEE/SMC International Conference on System of Systems Engineering*. Los Angeles, California, USA : IEEE, 2006, p. 41-46. isbn : 978-1-4244-0188-8. doi : 10.1109/SYSOSE.2006.1652271. url : http://ieeexplore.ieee.org/document/1652271/ (visité le 25/08/2022).

[2] Lorenzo Addazi, Antonio Cicchetti et Juri Di Rocco. "Semantic-based Model Matching with EMFCompare". en. In : ACM (2016), p. 10. doi : 10.1145/1235.

[3] Carlos Agostinho. "Information Models and Transformation Principles Applied to Servitization of Manufacturing and Service Systems Design :" en. In : *Proceedings of the 2nd International Conference on Model-Driven Engineering and Software Development*. Lisbon, Portugal : SCITEPRESS - Science, 2014, p. 657-665. isbn : 978-989-758-007-9. doi : 10.5220/0004875206570665. url : http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0004875206570665 (visité le 07/12/2020).

[4] Carlos Agostinho et al. "Towards a sustainable interoperability in networked enterprise information systems : Trends of knowledge and model-driven technology". en. In : *Computers in Industry* 79 (juin 2016), p. 64-76. issn : 01663615. doi : 10.1016/j.compind.2015.07.001. url : https://linkinghub.elsevier.com/retrieve/pii/S0166361515300191 (visité le 14/11/2022).

[5] Ahmed Ahmed, Mathias Kleiner et Lionel Roucoules. "Model-Based Interoperability IoT Hub for the Supervision of Smart Gas Distribution Networks". en. In : *IEEE Systems Journal* 13.2 (juin 2019), p. 1526-1533. issn : 1932-8184, 1937-9234, 2373-7816. doi : 10.1109/JSYST.2018.2851663. url : https://ieeexplore.ieee.org/document/8418391/ (visité le 15/02/2022).

[6] (Inria) AltanMod Research Group. *Class to relational transformation example*. Class to relational transformation example. url : https://www.eclipse.org/atl/atlTransformations/#Class2Relational..

[7] (Inria) AltanMod Research Group. *Family to person transformation example*. url : https://www.eclipse.org/atl/documentation/old/ATLUseCase_Families2Persons.pdf.

[8] Ali A. Alwan et al. "A Survey of Schema Matching Research using Database Schemas and Instances". en. In : *International Journal of Advanced Computer Science and Applications* 8.10 (2017). issn : 21565570, 2158107X. doi : 10.14569/IJACSA.2017.081014. url : http://thesai.org/Publications/ViewPaper?Volume=8&Issue=10&Code=ijacsa&SerialNo=14 (visité le 01/02/2022).

[9] Anam Amjad et al. “A Systematic Review on the Data Interoperability of Application Layer Protocols in Industrial IoT”. en. In : *IEEE Access* 9 (2021), p. 96528-96545. ISSN : 2169-3536. DOI : 10.1109/ACCESS.2021.3094763. URL : https://ieeexplore.ieee.org/document/9474488/ (visité le 23/05/2023).

[10] Ricardo Baeza-Yates, Berthier Ribeiro-Neto et al. *Modern information retrieval*. T. 463. 1999. ACM press New York, 1999.

[11] Dzmitry Bahdanau, Kyunghyun Cho et Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. en. arXiv :1409.0473 [cs, stat]. Mai 2016. URL : http://arxiv.org/abs/1409.0473 (visité le 21/08/2023).

[12] Chunguang Bai et al. “Industry 4.0 technologies assessment : A sustainability perspective”. en. In : *International Journal of Production Economics* 229 (nov. 2020), p. 107776. ISSN : 09255273. DOI : 10.1016/j.ijpe.2020.107776. URL : https://linkinghub.elsevier.com/retrieve/pii/S0925527320301559 (visité le 05/07/2023).

[13] Islem Baki et Houari Sahraoui. “Multi-Step Learning and Adaptive Search for Learning Complex Model Transformations from Examples”. In : *ACM Transactions on Software Engineering and Methodology* 25.3 (22 août 2016), p. 1-37. ISSN : 1049-331X, 1557-7392. DOI : 10.1145/2904904. URL : https://dl.acm.org/doi/10.1145/2904904 (visité le 20/04/2023).

[14] Zoltán Balogh et Dániel Varró. “Model transformation by example using inductive logic programming”. In : *Software & Systems Modeling* 8.3 (juill. 2009), p. 347-364. ISSN : 1619-1366, 1619-1374. DOI : 10.1007/s10270-008-0092-1. URL : http://link.springer.com/10.1007/s10270-008-0092-1 (visité le 20/04/2023).

[15] Angela Barriga, Adrian Rutle et Rogardt Heldal. “Personalized and Automatic Model Repairing using Reinforcement Learning”. en. In : *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. Munich, Germany : IEEE, sept. 2019, p. 175-181. ISBN : 978-1-72815-125-0. DOI : 10.1109/MODELS-C.2019.00030. URL : https://ieeexplore.ieee.org/document/8904758/ (visité le 06/09/2021).

[16] Alexandre Bento, Amal Zouaq et Michel Gagnon. “Ontology Matching Using Convolutional Neural Networks”. en. In : Marseille, France : European Language Resources Association, 2020, p. 5648-5653. URL : https://aclanthology.org/2020.lrec-1.693.

[17] Dirk Beyer, Claus Lewerentz et Frank Simon. “Impact of Inheritance on Metrics for Size, Coupling, and Cohesion in Object-Oriented Systems”. en. In : *New Approaches in Software Measurement*. Sous la dir. de Gerhard Goos et al. T. 2006. Series Title : Lecture Notes in Computer Science. Berlin, Heidelberg : Springer Berlin Heidelberg, 2001, p. 1-17. ISBN : 978-3-540-41727-9 978-3-540-44704-7. DOI : 10.1007/3-540-44704-0_1. URL : http://link.springer.com/10.1007/3-540-44704-0_1 (visité le 23/08/2021).

[18] Jean Bézivin. “On the unification power of models”. en. In : *Software & Systems Modeling* 4.2 (mai 2005), p. 171-188. ISSN : 1619-1366, 1619-1374. DOI : 10.1007/s10270-005-0079-0. URL : http://link.springer.com/10.1007/s10270-005-0079-0 (visité le 24/04/2023).

[19] Jean Bézivin et al. “Model Transformations ? Transformation Models !” In : *Model Driven Engineering Languages and Systems*. Sous la dir. de David Hutchison et al. T. 4199. Series Title : Lecture Notes in Computer Science. Berlin, Heidelberg : Springer Berlin Heidelberg, 2006, p. 440-453. ISBN : 978-3-540-45772-5 978-3-540-45773-2. DOI : 10.1007/11880240_31. URL : http://link.springer.com/10.1007/11880240_31 (visité le 03/05/2023).

[20] Fredrik Blomstedt et al. “The arrowhead approach for SOA application development and documentation”. en. In : *IECON 2014 - 40th Annual Conference of the IEEE Industrial Electronics Society*. Dallas, TX, USA : IEEE, oct. 2014, p. 2631-2637. isbn : 978-1-4799-4032-5. doi : 10.1109/IECON.2014.7048877. url : http://ieeexplore.ieee.org/document/7048877/ (visité le 23/08/2023).

[21] J. Boardman et B. Sauser. “System of Systems - the meaning of of”. en. In : *2006 IEEE/SMC International Conference on System of Systems Engineering*. Los Angeles, California, USA : IEEE, 2006, p. 118-123. isbn : 978-1-4244-0188-8. doi : 10.1109/SYSOSE.2006.1652284. url : http://ieeexplore.ieee.org/document/1652284/ (visité le 22/08/2022).

[22] Antoine Bordes et al. “Translating Embeddings for Modeling Multi-relational Data”. en. In : *Advances in Neural Information Processing Systems 26* (2013), p. 9. issn : 9781632660244.

[23] Carsten Bormann, Angelo P. Castellani et Zach Shelby. “CoAP : An Application Protocol for Billions of Tiny Internet Nodes”. en. In : *IEEE Internet Computing* 16.2 (mars 2012), p. 62-67. issn : 1089-7801. doi : 10.1109/MIC.2012.29. url : http://ieeexplore.ieee.org/document/6159216/ (visité le 19/06/2023).

[24] Jean-Pierre Bourey et al. *Report on Model Driven Interoperability*. 2007.

[25] Alexandros Bousdekis et al. “A Review of Data-Driven Decision-Making Methods for Industry 4.0 Maintenance Applications”. en. In : *Electronics* 10.7 (mars 2021), p. 828. issn : 2079-9292. doi : 10.3390/electronics10070828. url : https://www.mdpi.com/2079-9292/10/7/828 (visité le 09/06/2023).

[26] Juliana Bowles et Dulani Meedeniya. “Formal Transformation from Sequence Diagrams to Coloured Petri Nets”. en. In : *2010 Asia Pacific Software Engineering Conference*. Sydney, Australia : IEEE, nov. 2010, p. 216-225. isbn : 978-1-4244-8831-5. doi : 10.1109/APSEC.2010.33. url : http://ieeexplore.ieee.org/document/5693197/ (visité le 20/09/2023).

[27] Tim Bray et al. “Extensible Markup Language (XML) 1.1 (Second Edition)”. en. In : ().

[28] Malte Brettel et al. “How Virtualization, Decentralization and Network Building Change the Manufacturing Landscape : An Industry 4.0 Perspective”. en. In : *International Journal of Information and Communication Engineering* 8.1 (2014), p. 37-44.

[29] Q Brilhault, E Yahia et L Roucoules. “Digital continuity based on reinforcement learning model transformations”. en. In : Ischia, Italie : Lectures Notes in Mechanical Engineering, Springer, juin 2022, p. 12. doi : 10.1007/978-3-031-15928-2_39.

[30] Jean-Michel Bruel et al. “Comparing and classifying model transformation reuse approaches across metamodels”. en. In : *Software and Systems Modeling* 19.2 (mars 2020), p. 441-465. issn : 1619-1366, 1619-1374. doi : 10.1007/s10270-019-00762-9. url : http://link.springer.com/10.1007/s10270-019-00762-9 (visité le 03/05/2023).

[31] Adauto Bueno, Moacir Godinho Filho et Alejandro G. Frank. “Smart production planning and control in the Industry 4.0 context : A systematic literature review”. en. In : *Computers & Industrial Engineering* 149 (nov. 2020), p. 106774. issn : 03608352. doi : 10.1016/j.cie.2020.106774. url : https://linkinghub.elsevier.com/retrieve/pii/S0360835220304861 (visité le 09/06/2023).

[32] Loli Burgueno, Jordi Cabot et Sebastien Gerard. “An LSTM-Based Neural Network Architecture for Model Transformations”. In : *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS)*. 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS). Munich, Germany : IEEE, sept. 2019, p. 294-299. isbn : 978-1-72812-536-7. doi : 10.1109/MODELS.2019.00013. url : https://ieeexplore.ieee.org/document/8906971/ (visité le 20/04/2023).

[33] Loli Burgueño et al. “A generic LSTM neural network architecture to infer heterogeneous model transformations”. In : *Software and Systems Modeling* 21.1 (fév. 2022), p. 139-156. issn : 1619-1366, 1619-1374. doi : 10.1007/s10270-021-00893-y. url : https://link.springer.com/10.1007/s10270-021-00893-y (visité le 20/04/2023).

[34] Jordi Cabot et al. “Cognifying Model-Driven Software Engineering”. en. In : *Software Technologies : Applications and Foundations*. Sous la dir. de Martina Seidl et Steffen Zschaler. T. 10748. Series Title : Lecture Notes in Computer Science. Cham : Springer International Publishing, 2018, p. 154-160. isbn : 978-3-319-74729-3 978-3-319-74730-9. doi : 10.1007/978-3-319-74730-9_13. url : http://link.springer.com/10.1007/978-3-319-74730-9_13 (visité le 06/09/2021).

[35] David Chen. “Framework for Enterprise Interoperability”. en. In : *Enterprise Interoperability : INTEROP-PGSO Vision* 1 (mai 2017). doi : https://doi.org/10.1002/9781119407928.ch1.

[36] David Chen, Guy Doumeingts et François Vernadat. “Architectures for enterprise integration and interoperability : Past, present and future”. en. In : *Computers in Industry* 59.7 (sept. 2008), p. 647-659. issn : 01663615. doi : 10.1016/j.compind.2007.12.016. url : https://linkinghub.elsevier.com/retrieve/pii/S0166361508000365 (visité le 17/08/2021).

[37] David Chen et François B. Vernadat. “Enterprise Interoperability : A Standardisation View”. en. In : *Enterprise Inter- and Intra-Organizational Integration*. T. 108. Series Title : IFIP Advances in Information and Communication Technology. Boston, MA : Springer US, 2003, p. 273-282. isbn : 978-1-4757-5151-2 978-0-387-35621-1. doi : 10.1007/978-0-387-35621-1_28. url : http://link.springer.com/10.1007/978-0-387-35621-1_28 (visité le 12/06/2023).

[38] Min Chen, Shiwen Mao et Yunhao Liu. “Big Data : A Survey”. en. In : *Mobile Networks and Applications* 19.2 (avr. 2014), p. 171-209. issn : 1383-469X, 1572-8153. doi : 10.1007/s11036-013-0489-0. url : http://link.springer.com/10.1007/s11036-013-0489-0 (visité le 19/08/2022).

[39] Xiaojun Chen, Shengbin Jia et Yang Xiang. “A review : Knowledge reasoning over knowledge graph”. en. In : *Expert Systems with Applications* 141 (mars 2020), p. 112948. issn : 09574174. doi : 10.1016/j.eswa.2019.112948. url : https://linkinghub.elsevier.com/retrieve/pii/S0957417419306669 (visité le 07/02/2021).

[40] Shyam R Chidamber. “A metrics suite for object oriented design”. In : *IEEE Transactions on Software Engineering* (1994), p. 50. issn : 0098-5589. doi : 10.1109/32.295895. url : https://ieeexplore.ieee.org/abstract/document/295895.

[41] Chiara CIMINI et al. "Industry 4.0 Technologies Impacts in the Manufacturing and Supply Chain Landscape : An Overview". en. In : *Service Orientation in Holonic and Multi-Agent Manufacturing*. T. 803. Cham : Springer International Publishing, 2019, p. 109-120. ISBN : 978-3-030-03002-5 978-3-030-03003-2. DOI : 10.1007/978-3-030-03003-2_8. URL : http://link.springer.com/10.1007/978-3-030-03003-2_8 (visité le 31/08/2022).

[42] Anne M. CREGAN. "Symbol Grounding for the Semantic Web". en. In : *The Semantic Web : Research and Applications*. Sous la dir. d'Enrico FRANCONI, Michael KIFER et Wolfgang MAY. T. 4519. ISSN : 0302-9743, 1611-3349 Series Title : Lecture Notes in Computer Science. Berlin, Heidelberg : Springer Berlin Heidelberg, 2007, p. 429-442. ISBN : 978-3-540-72666-1 978-3-540-72667-8. DOI : 10.1007/978-3-540-72667-8_31. URL : http://link.springer.com/10.1007/978-3-540-72667-8_31 (visité le 19/08/2021).

[43] Isabel F CRUZ et Huiyong XIAO. "The Role of Ontologies in Data Integration". en. In : *Engineering intelligent systems for electrical engineering and communications* 13.4 (2005), p. 245.

[44] Gyorgy CSERTAN et al. "VIATRA - Visual Automated Transformations for Formal Verification and Validation of UML Models (Tool demonstration)". en. In : Edinburgh, UK, 2002. ISBN : 0-7695-1736-6. DOI : 10.1109/ASE.2002.1115027.

[45] Peng CUI et al. *A Survey on Network Embedding*. arXiv :1711.08752 [cs]. Nov. 2017. URL : http://arxiv.org/abs/1711.08752 (visité le 06/07/2023).

[46] K. CZARNECKI et S. HELSEN. "Feature-based survey of model transformation approaches". In : *IBM Systems Journal* 45.3 (2006), p. 621-645. ISSN : 0018-8670. DOI : 10.1147/sj.453.0621. URL : http://ieeexplore.ieee.org/document/5386627/ (visité le 20/04/2023).

[47] Krzysztof CZARNECKI et Simon HELSEN. "Classification of Model Transformation Approaches". en. In : t. 45. 2003, p. 1-17.

[48] Wenbin DAI et al. "Semantic Integration of Plug-and-Play Software Components for Industrial Edges Based on Microservices". en. In : *IEEE Access* 7 (2019), p. 125882-125892. ISSN : 2169-3536. DOI : 10.1109/ACCESS.2019.2938565. URL : https://ieeexplore.ieee.org/document/8821289/ (visité le 24/05/2023).

[49] Jerker DELSING. *Iot automation : Arrowhead framework*. Crc Press, 2017.

[50] Hasan DERHAMY et al. "Translation error handling for multi-protocol SOA systems". en. In : *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*. Luxembourg, Luxembourg : IEEE, sept. 2015, p. 1-8. ISBN : 978-1-4673-7929-8. DOI : 10.1109/ETFA.2015.7301473. URL : http://ieeexplore.ieee.org/document/7301473/ (visité le 28/08/2023).

[51] Jacob DEVLIN et al. "BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding". en. In : *arXiv :1810.04805 [cs]* (mai 2019). arXiv : 1810.04805. URL : http://arxiv.org/abs/1810.04805 (visité le 16/08/2021).

[52] Xavier DOLQUES et al. "Easing Model Transformation Learning with Automatically Aligned Examples". In : *Modelling Foundations and Applications*. T. 6698. Series Title : Lecture Notes in Computer Science. Berlin, Heidelberg : Springer Berlin Heidelberg, 2011, p. 189-204. ISBN : 978-3-642-21469-1 978-3-642-21470-7. DOI : 10.1007/978-3-642-21470-7_14. URL : http://link.springer.com/10.1007/978-3-642-21470-7_14 (visité le 20/04/2023).

[53] Xavier 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*. 2010 14th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW). Vit ria, Brazil : IEEE, oct. 2010, p. 27-32. isbn : 978-1-4244-7965-8. doi : 10.1109/EDOCW.2010.32. url : http://ieeexplore.ieee.org/document/5628955/ (visité le 20/04/2023).

[54] Ali Dorri, Salil S. Kanhere et Raja Jurdak. “Multi-Agent Systems : A Survey”. en. In : *IEEE Access* 6 (2018), p. 28573-28593. issn : 2169-3536. doi : 10.1109/ACCESS.2018.2831228. url : https://ieeexplore.ieee.org/document/8352646/ (visité le 29/08/2023).

[55] Nicola Dragoni et al. *Microservices : yesterday, today, and tomorrow*. en. arXiv :1606.04036 [cs]. Avr. 2017. url : http://arxiv.org/abs/1606.04036 (visité le 23/08/2023).

[56] EC. “Enterprise Interoperability research roadmap”. en. In : (2006).

[57] Martin Eisenberg et al. “Towards Reinforcement Learning for In-Place Model Transformations”. en. In : *2021 ACM/IEEE 24th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. Fukuoka, Japan : IEEE, oct. 2021, p. 82-88. isbn : 978-1-66543-495-9. doi : 10.1109/MODELS50736.2021.00017. url : https://ieeexplore.ieee.org/document/9592463/ (visité le 17/01/2022).

[58] Jean-Rémy Falleri et al. “Metamodel Matching for Automatic Model Transformation Generation”. en. In : *Model Driven Engineering Languages and Systems*. T. 5301. ISSN : 0302-9743, 1611-3349 Series Title : Lecture Notes in Computer Science. Berlin, Heidelberg : Springer Berlin Heidelberg, 2008, p. 326-340. isbn : 978-3-540-87874-2 978-3-540-87875-9. doi : 10.1007/978-3-540-87875-9_24. url : http://link.springer.com/10.1007/978-3-540-87875-9_24 (visité le 24/08/2021).

[59] Shichao Fang et Kevin Lano. “Extracting Correspondences from Metamodels Using Metamodel Matching”. en. In : STAF (Co-Located Events) (2019), p. 3-8.

[60] Martin Faunes, Houari Sahraoui et Mounir Boukadoum. “Generating model transformation rules from examples using an evolutionary algorithm”. In : *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*. ASE’12 : IEEE/ACM International Conference on Automated Software Engineering. Essen Germany : ACM, 3 sept. 2012, p. 250-253. isbn : 978-1-4503-1204-2. doi : 10.1145/2351676.2351714. url : https://dl.acm.org/doi/10.1145/2351676.2351714 (visité le 20/04/2023).

[61] Martin Faunes, Houari Sahraoui et Mounir Boukadoum. “Genetic-Programming Approach to Learn Model Transformation Rules from Examples”. In : *Theory and Practice of Model Transformations*. T. 7909. Series Title : Lecture Notes in Computer Science. Berlin, Heidelberg : Springer Berlin Heidelberg, 2013, p. 17-32. isbn : 978-3-642-38882-8 978-3-642-38883-5. doi : 10.1007/978-3-642-38883-5_2. url : http://link.springer.com/10.1007/978-3-642-38883-5_2 (visité le 20/04/2023).

[62] Juliana Fernandes et al. “How can interoperability approaches impact on Systems-of-Information Systems characteristics ?” en. In : *XVI Brazilian Symposium on Information Systems*. São Bernardo do Campo Brazil : ACM, nov. 2020, p. 1-8. isbn : 978-1-4503-8873-3. doi : 10.1145/3411564.3411621. url : https://dl.acm.org/doi/10.1145/3411564.3411621 (visité le 19/08/2022).

[63] Alejandro G. Frank et al. “Servitization and Industry 4.0 convergence in the digital transformation of product firms : A business model innovation perspective”. en. In : *Technological Forecasting and Social Change* 141 (avr. 2019), p. 341-351. issn : 00401625. doi : 10.1016/j.techfore.2019.01.014. url : https://linkinghub.elsevier.com/retrieve/pii/S0040162518311156 (visité le 12/06/2023).

[64] Maria Ganzha et al. “Streaming semantic translations”. en. In : *2017 21st International Conference on System Theory, Control and Computing (ICSTCC)*. Sinaia : IEEE, oct. 2017, p. 1-8. isbn : 978-1-5386-3842-2. doi : 10.1109/ICSTCC.2017.8107003. url : http://ieeexplore.ieee.org/document/8107003/ (visité le 10/09/2021).

[65] Morteza Ghobakhloo. “Industry 4.0, digitization, and opportunities for sustainability”. en. In : *Journal of Cleaner Production* 252 (avr. 2020), p. 119869. issn : 09596526. doi : 10.1016/j.jclepro.2019.119869. url : https://linkinghub.elsevier.com/retrieve/pii/S0959652619347390 (visité le 10/11/2022).

[66] Justin Gilmer et al. “Neural Message Passing for Quantum Chemistry”. en. In : t. 70. 2017, p. 1263-1272. url : https://proceedings.mlr.press/v70/gilmer17a.html.

[67] Paul Grace, Brian Pickering et Mike Surridge. “Model-driven interoperability : engineering heterogeneous IoT systems”. en. In : *Annals of Telecommunication* (2015), p. 10. url : https://link.springer.com/article/10.1007/s12243-015-0487-2.

[68] Roy Grønmo et Birger Møller-Pedersen. “From UML 2 Sequence Diagrams to State Machines by Graph Transformation.” en. In : *The Journal of Object Technology* 10 (2011), 8 :1. issn : 1660-1769. doi : 10.5381/jot.2011.10.1.a8. url : http://www.jot.fm/contents/issue_2011_01/article8.html (visité le 20/09/2023).

[69] Xiaojie Guo, Shiyu Wang et Liang Zhao. “Graph Neural Networks : Graph Transformation”. en. In : *Graph Neural Networks : Foundations, Frontiers, and Applications*. Sous la dir. de Lingfei Wu et al. Singapore : Springer Nature Singapore, 2022, p. 251-275. isbn : 9789811660535 9789811660542. doi : 10.1007/978-981-16-6054-2_12. url : https://link.springer.com/10.1007/978-981-16-6054-2_12 (visité le 21/08/2023).

[70] Didem Gürdür et Fredrik Asplund. “A systematic review to merge discourses : Interoperability, integration and cyber-physical systems”. en. In : *Journal of Industrial Information Integration* 9 (mars 2018), p. 14-23. issn : 2452414X. doi : 10.1016/j.jii.2017.12.001. url : https://linkinghub.elsevier.com/retrieve/pii/S2452414X17300687 (visité le 24/08/2021).

[71] Amelie Gyrard et Martin Serrano. “A Unified Semantic Engine for Internet of Things and Smart Cities : From Sensor Data to End-Users Applications”. en. In : *2015 IEEE International Conference on Data Science and Data Intensive Systems*. Sydney, Australia : IEEE, déc. 2015, p. 718-725. isbn : 978-1-5090-0214-6. doi : 10.1109/DSDIS.2015.59. url : http://ieeexplore.ieee.org/document/7396579/ (visité le 19/08/2021).

[72] Amelie Gyrard, Antoine Zimmermann et Amit Sheth. “Building IoT-Based Applications for Smart Cities : How Can Ontology Catalogs Help ?” en. In : *IEEE Internet of Things Journal* 5.5 (oct. 2018), p. 3978-3990. issn : 2327-4662, 2372-2541. doi : 10.1109/JIOT.2018.2854278. url : https://ieeexplore.ieee.org/document/8409270/ (visité le 19/08/2021).

[73] Will Hamilton, Zhitao Ying et Jure Leskovec. “Inductive Representation Learning on Large Graphs”. en. In : *Proceedings of the 31st International Conference on Neural Information Processing Systems* (2017), p. 1025-1035.

[74] William L Hamilton. *Graph Representation Learning*. en. Morgan & Claypool Publishers. T. 46. 2020. isbn : 1-68173-963-1 978-1-68173-963-2.

[75] William L. Hamilton, Rex Ying et Jure Leskovec. *Representation Learning on Graphs : Methods and Applications*. en. arXiv :1709.05584 [cs]. Avr. 2018. url : http://arxiv.org/abs/1709.05584 (visité le 06/07/2023).

[76] Junheng Hao et al. “MEDTO : Medical Data to Ontology Matching Using Hybrid Graph Neural Networks”. en. In : *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. Virtual Event Singapore : ACM, août 2021, p. 2946-2954. isbn : 978-1-4503-8332-5. doi : 10.1145/3447548.3467138. url : https://dl.acm.org/doi/10.1145/3447548.3467138 (visité le 11/07/2023).

[77] Stevan Harnad. “THE SYMBOL GROUNDING PROBLEM”. en. In : *Elsevier* 42 (1990), p. 335-346. doi : https://doi.org/10.1016/0167-2789(90)90087-6.

[78] SANDRA HEILER. “Semantic interoperability”. en. In : *ACM Computmg Surveys* 27.2 (1995).

[79] Mario Hermann, Tobias Pentek et Boris Otto. “Design principles for Industrie 4.0 scenarios : a literature review”. In : t. 45. Koloa, HI, USA : IEEE, 2016. doi : 10.1109/HICSS.2016.488.

[80] Alan Hevner et Samir Chatterjee. “Design Science Research in Information Systems”. en. In : *Design Research in Information Systems*. T. 22. Series Title : Integrated Series in Information Systems. Boston, MA : Springer US, 2010, p. 9-22. isbn : 978-1-4419-5652-1 978-1-4419-5653-8. doi : 10.1007/978-1-4419-5653-8_2. url : http://link.springer.com/10.1007/978-1-4419-5653-8_2 (visité le 06/06/2023).

[81] M. Huchard et al. “Relational concept discovery in structured datasets”. In : *Annals of Mathematics and Artificial Intelligence* 49.1 (2 août 2007), p. 39-76. issn : 1012-2443, 1573-7470. doi : 10.1007/s10472-007-9056-3. url : http://link.springer.com/10.1007/s10472-007-9056-3 (visité le 21/04/2023).

[82] Ryutaro Ichise. “Machine Learning Approach for Ontology Mapping Using Multiple Concept Similarity Measures”. en. In : *Seventh IEEE/ACIS International Conference on Computer and Information Science (icis 2008)*. IEEE, mai 2008, p. 340-346. isbn : 978-0-7695-3131-1. doi : 10.1109/ICIS.2008.51. url : http://ieeexplore.ieee.org/document/4529843/ (visité le 11/07/2023).

[83] “IEEE Standard Computer Dictionary : A Compilation of IEEE Standard Computer Glossaries”. In : *IEEE Std 610* (1991), p. 1-217. doi : 10.1109/IEEESTD.1991.106963. url : https://ieeexplore.ieee.org/document/182763.

[84] Mehdi Iraqi-Houssaini, Mathias Kleiner et Lionel Roucoules. “Model-Based (Mechanical) Product Design”. en. In : *Model Driven Engineering Languages and Systems*. Sous la dir. de Jon Whittle, Tony Clark et Thomas Kühne. T. 6981. Series Title : Lecture Notes in Computer Science. Berlin, Heidelberg : Springer Berlin Heidelberg, 2011, p. 548-562. isbn : 978-3-642-24484-1 978-3-642-24485-8. doi : 10.1007/978-3-642-24485-8_40. url : http://link.springer.com/10.1007/978-3-642-24485-8_40 (visité le 11/02/2021).

[85] Vivek Iyer, Arvind Agarwal et Harshit Kumar. "VeeAlign : Multifaceted Context Representation using Dual Attention for Ontology Alignment". In : (2021). Publisher : arXiv Version Number : 3. doi : 10.48550/ARXIV.2102.04081. url : https://arxiv.org/abs/2102.04081 (visité le 11/07/2023).

[86] Vaclav Jirkovsky, Marek Obitko et Vladimir Marik. "Understanding Data Heterogeneity in the Context of Cyber-Physical Systems Integration". en. In : *IEEE Transactions on Industrial Informatics* 13.2 (avr. 2017), p. 660-667. issn : 1551-3203, 1941-0050. doi : 10.1109/TII.2016.2596101. url : http://ieeexplore.ieee.org/document/7524742/ (visité le 24/05/2023).

[87] Vaclav Jirkovsky et al. "Toward Plug&Play Cyber-Physical System Components". en. In : *IEEE Transactions on Industrial Informatics* 14.6 (juin 2018), p. 2803-2811. issn : 1551-3203, 1941-0050. doi : 10.1109/TII.2018.2794982. url : https://ieeexplore.ieee.org/document/8263185/ (visité le 24/05/2023).

[88] Frédéric Jouault et Ivan Kurtev. "Transforming Models with ATL". en. In : *Satellite Events at the MoDELS 2005 Conference*. T. 3844. Series Title : Lecture Notes in Computer Science. Berlin, Heidelberg : Springer Berlin Heidelberg, 2006, p. 128-138. isbn : 978-3-540-31780-7 978-3-540-31781-4. doi : 10.1007/11663430_14. url : http://link.springer.com/10.1007/11663430_14 (visité le 17/08/2021).

[89] Nafiseh Kahani et al. "Survey and classification of model transformation tools". en. In : *Software & Systems Modeling* 18.4 (août 2019), p. 2361-2397. issn : 1619-1366, 1619-1374. doi : 10.1007/s10270-018-0665-6. url : http://link.springer.com/10.1007/s10270-018-0665-6 (visité le 03/05/2023).

[90] Hung-An Kao et al. "A Cyber Physical Interface for Automation Systems—Methodology and Examples". en. In : *Machines* 3.2 (mai 2015), p. 93-106. issn : 2075-1702. doi : 10.3390/machines3020093. url : http://www.mdpi.com/2075-1702/3/2/93 (visité le 22/08/2022).

[91] Marouane Kessentini, Houari Sahraoui et Mounir Boukadoum. "Model Transformation as an Optimization Problem". In : *Model Driven Engineering Languages and Systems*. T. 5301. ISSN : 0302-9743, 1611-3349 Series Title : Lecture Notes in Computer Science. Berlin, Heidelberg : Springer Berlin Heidelberg, 2008, p. 159-173. isbn : 978-3-540-87874-2 978-3-540-87875-9. doi : 10.1007/978-3-540-87875-9_12. url : http://link.springer.com/10.1007/978-3-540-87875-9_12 (visité le 20/04/2023).

[92] Marouane Kessentini et al. "Search-based metamodel matching with structural and syntactic measures". en. In : *Journal of Systems and Software* 97 (nov. 2014), p. 1-14. issn : 01641212. doi : 10.1016/j.jss.2014.06.040. url : https://linkinghub.elsevier.com/retrieve/pii/S0164121214001484 (visité le 16/06/2021).

[93] Marouane Kessentini et al. "Search-based model transformation by example". In : *Software & Systems Modeling* 11.2 (mai 2012), p. 209-226. issn : 1619-1366, 1619-1374. doi : 10.1007/s10270-010-0175-7. url : http://link.springer.com/10.1007/s10270-010-0175-7 (visité le 20/04/2023).

[94] Maqbool Khan et al. "Big data challenges and opportunities in the hype of Industry 4.0". en. In : *2017 IEEE International Conference on Communications (ICC)*. Paris, France : IEEE, mai 2017, p. 1-6. isbn : 978-1-4673-8999-0. doi : 10.1109/ICC.2017.7996801. url : http://ieeexplore.ieee.org/document/7996801/ (visité le 14/06/2023).

[95] Thomas N. Kipf et Max Welling. “Semi-Supervised Classification with Graph Convolutional Networks”. en. In : *arXiv :1609.02907 [cs, stat]* (fév. 2017). arXiv : 1609.02907. URL : http://arxiv.org/abs/1609.02907 (visité le 08/11/2021).

[96] Anneke Kleppe, Jos Warmer et Wim Bast. “The Model Driven Architecture™ : Practice and Promise”. en. In : *Addison-Wesley Professional* (2003). ISSN : 9780321194428.

[97] Christos Koutras et al. *SiMa : Effective and Efficient Data Silo Federation Using Graph Neural Networks*. en. arXiv :2206.12733 [cs]. Juin 2022. URL : http://arxiv.org/abs/2206.12733 (visité le 21/08/2023).

[98] Daniel Kozma, Pal Varga et Gabor Soos. “Supporting Digital Production, Product Lifecycle and Supply Chain Management in Industry 4.0 by the Arrowhead Framework – a Survey”. en. In : *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*. Helsinki, Finland : IEEE, juill. 2019, p. 126-131. ISBN : 978-1-72812-927-3. DOI : 10.1109/INDIN41052.2019.8972216. URL : https://ieeexplore.ieee.org/document/8972216/ (visité le 23/08/2023).

[99] A. Kusel et al. “Reuse in model-to-model transformation languages : are we there yet ?” en. In : *Software & Systems Modeling* 14.2 (mai 2015), p. 537-572. ISSN : 1619-1366, 1619-1374. DOI : 10.1007/s10270-013-0343-7. URL : http://link.springer.com/10.1007/s10270-013-0343-7 (visité le 03/05/2023).

[100] Angelika Kusel et al. “A Survey on Incremental Model Transformation Approaches”. en. In : *MoDELS 2013 Workshops* 1090 (2013), p. 4-13. DOI : CEUR-WS.org.

[101] K. Lano et al. “Enhancing model transformation synthesis using natural language processing”. en. In : *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems : Companion Proceedings*. Virtual Event Canada : ACM, oct. 2020, p. 1-10. ISBN : 978-1-4503-8135-2. DOI : 10.1145/3417990.3421386. URL : https://dl.acm.org/doi/10.1145/3417990.3421386 (visité le 19/05/2023).

[102] Kevin Lano et Shichao Fang. “Automated Synthesis of ATL Transformations from Metamodel Correspondences :” en. In : *Proceedings of the 8th International Conference on Model-Driven Engineering and Software Development*. Valletta, Malta : SCITEPRESS - Science et Technology Publications, 2020, p. 263-270. ISBN : 978-989-758-400-8. DOI : 10.5220/0008873702630270. URL : http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0008873702630270 (visité le 12/07/2021).

[103] Juan De Lara et Hans Vangheluwe. “AToM3 : A Tool for Multi-formalism and Metamodelling”. en. In : *Fundamental Approaches to Software Engineering*. Sous la dir. de Gerhard Goos et al. T. 2306. Series Title : Lecture Notes in Computer Science. Berlin, Heidelberg : Springer Berlin Heidelberg, 2002, p. 174-188. ISBN : 978-3-540-43353-8 978-3-540-45923-1. DOI : 10.1007/3-540-45923-5_12. URL : http://link.springer.com/10.1007/3-540-45923-5_12 (visité le 30/06/2023).

[104] Heiner Lasi et al. “Industry 4.0”. en. In : *Business & Information Systems Engineering* 6.4 (août 2014), p. 239-242. ISSN : 1867-0202. DOI : 10.1007/s12599-014-0334-4. URL : http://link.springer.com/10.1007/s12599-014-0334-4 (visité le 17/08/2022).

[105] Jay Lee, Behrad Bagheri et Hung-An Kao. “A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems”. en. In : *Manufacturing Letters* 3 (jan. 2015), p. 18-23. ISSN : 22138463. DOI : 10.1016/j.mfglet.2014.12.001. URL : https://linkinghub.elsevier.com/retrieve/pii/S221384631400025X (visité le 08/06/2023).

[106] Jens LEHMANN et al. "DBpedia – A large-scale, multilingual knowledge base extracted from Wikipedia". en. In : *Semantic Web* 6.2 (2015), p. 167-195. ISSN : 15700844. DOI : 10.3233/SW-140134. URL : https://www.medra.org/servlet/aliasResolver?alias=iospress&doi=10.3233/SW-140134 (visité le 06/07/2023).

[107] Francesco LELLI. "Interoperability of the Time of Industry 4.0 and the Internet of Things". en. In : *Future Internet* 11.2 (fév. 2019), p. 36. ISSN : 1999-5903. DOI : 10.3390/fi11020036. URL : http://www.mdpi.com/1999-5903/11/2/36 (visité le 31/08/2022).

[108] Grace A. LEWIS et al. "Why Standards Are Not Enough to Guarantee End-to-End Interoperability". en. In : *Seventh International Conference on Composition-Based Software Systems (ICCBSS 2008)*. Madrid, Spain : IEEE, fév. 2008, p. 164-173. ISBN : 978-0-7695-3091-8. DOI : 10.1109/ICCBSS.2008.25. URL : http://ieeexplore.ieee.org/document/4464021/ (visité le 02/09/2022).

[109] Jingran LI et al. "Big Data in product lifecycle management". en. In : *The International Journal of Advanced Manufacturing Technology* 81.1-4 (oct. 2015), p. 667-684. ISSN : 0268-3768, 1433-3015. DOI : 10.1007/s00170-015-7151-x. URL : http://link.springer.com/10.1007/s00170-015-7151-x (visité le 09/06/2023).

[110] Yingming LI, Ming YANG et Zhongfei ZHANG. "A Survey of Multi-View Representation Learning". en. In : *IEEE Transactions on Knowledge and Data Engineering* 31.10 (oct. 2019). arXiv :1610.01206 [cs], p. 1863-1883. ISSN : 1041-4347, 1558-2191, 2326-3865. DOI : 10.1109/TKDE.2018.2872063. URL : http://arxiv.org/abs/1610.01206 (visité le 07/07/2023).

[111] Chao LIU et Pingyu JIANG. "A Cyber-physical System Architecture in Shop Floor for Intelligent Manufacturing". en. In : *Procedia CIRP* 56 (2016), p. 372-377. ISSN : 22128271. DOI : 10.1016/j.procir.2016.10.059. URL : https://linkinghub.elsevier.com/retrieve/pii/S2212827116310514 (visité le 04/10/2023).

[112] Zhiyuan LIU et al. "Exploring and Evaluating Attributes, Values, and Structures for Entity Alignment". en. In : *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online : Association for Computational Linguistics, 2020, p. 6355-6364. DOI : 10.18653/v1/2020.emnlp-main.515. URL : https://www.aclweb.org/anthology/2020.emnlp-main.515 (visité le 23/07/2021).

[113] Denivaldo LOPES, Slimane HAMMOUDI et Zair ABDELOUAHAB. "Schema Matching in the Context of Model Driven Engineering : From Theory to Practice". In : *Advances in Systems, Computing Sciences and Software Engineering*. Dordrecht : Springer Netherlands, 2006, p. 219-227. ISBN : 978-1-4020-5263-7.

[114] Wolfgang MAHNKE, Stefan-Helmut LEITNER et Matthias DAMM. *OPC unified architecture*. en. OCLC : ocn268784080. Berlin : Springer, 2009. ISBN : 978-3-540-68898-3 978-3-540-68899-0.

[115] Mark W. MAIER. "Architecting principles for systems-of-systems". In : *Systems Engineering : The Journal of the International Council on Systems Engineering* 1.4 (1998), p. 267-284.

[116] Mohammed MAREE et Mohammed BELKHATIR. "Addressing semantic heterogeneity through multiple knowledge base assisted merging of domain-specific ontologies". en. In : *Knowledge-Based Systems* 73 (jan. 2015), p. 199-211. ISSN : 09507051. DOI : 10.1016/j.knosys.2014.10.001. URL : https://linkinghub.elsevier.com/retrieve/pii/S0950705114003682 (visité le 25/08/2021).

[117] Nicola Melluso, Irlan Grangel-González et Gualtiero Fantoni. “Enhancing Industry 4.0 standards interoperability via knowledge graphs with natural language processing”. en. In : *Computers in Industry* 140 (sept. 2022), p. 103676. ISSN : 01663615. DOI : 10.1016/j.compind.2022.103676. URL : https://linkinghub.elsevier.com/retrieve/pii/S0166361522000732 (visité le 15/11/2022).

[118] Tomas Mikolov et al. “Advances in Pre-Training Distributed Word Representations”. en. In : *arXiv :1712.09405 [cs]* (déc. 2017). arXiv : 1712.09405. URL : http://arxiv.org/abs/1712.09405 (visité le 16/08/2021).

[119] Tomas Mikolov et al. “Distributed Representations of Words and Phrases and their Compositionality”. en. In : *Advances in neural information processing systems* 26 (2013).

[120] George A. Miller. “WordNet : a lexical database for English”. en. In : *Communications of the ACM* 38.11 (nov. 1995), p. 39-41. ISSN : 0001-0782, 1557-7317. DOI : 10.1145/219717.219748. URL : https://dl.acm.org/doi/10.1145/219717.219748 (visité le 30/06/2023).

[121] Volodymyr Mnih et al. *Playing Atari with Deep Reinforcement Learning*. en. arXiv :1312.5602 [cs]. Déc. 2013. URL : http://arxiv.org/abs/1312.5602 (visité le 13/09/2022).

[122] G. E. Modoni et al. “Enhancing factory data integration through the development of an ontology : from the reference models reuse to the semantic conversion of the legacy models”. en. In : *International Journal of Computer Integrated Manufacturing* 30.10 (oct. 2017), p. 1043-1059. ISSN : 0951-192X, 1362-3052. DOI : 10.1080/0951192X.2016.1268720. URL : https://www.tandfonline.com/doi/full/10.1080/0951192X.2016.1268720 (visité le 25/04/2022).

[123] Alexandre Moeuf et al. “The industrial management of SMEs in the era of Industry 4.0”. en. In : *International Journal of Production Research* 56.3 (fév. 2018), p. 1118-1136. ISSN : 0020-7543, 1366-588X. DOI : 10.1080/00207543.2017.1372647. URL : https://www.tandfonline.com/doi/full/10.1080/00207543.2017.1372647 (visité le 12/06/2023).

[124] L. Monostori et al. “Cyber-physical systems in manufacturing”. en. In : *CIRP Annals* 65.2 (2016), p. 621-641. ISSN : 00078506. DOI : 10.1016/j.cirp.2016.06.005. URL : https://linkinghub.elsevier.com/retrieve/pii/S0007850616301974 (visité le 09/06/2023).

[125] Claus Ballegaard Nielsen et al. “Systems of Systems Engineering : Basic Concepts, Model-Based Techniques, and Research Directions”. en. In : *ACM Computing Surveys* 48.2 (nov. 2015), p. 1-41. ISSN : 0360-0300, 1557-7341. DOI : 10.1145/2794381. URL : https://dl.acm.org/doi/10.1145/2794381 (visité le 22/08/2022).

[126] Jacob Nilsson, Fredrik Sandin et Jerker Delsing. “Interoperability and machine-to-machine translation model with mappings to machine learning tasks”. en. In : *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*. Helsinki, Finland : IEEE, juill. 2019, p. 284-289. ISBN : 978-1-72812-927-3. DOI : 10.1109/INDIN41052.2019.8972085. URL : https://ieeexplore.ieee.org/document/8972085/ (visité le 28/01/2021).

[127] Mahda Noura, Mohammed Atiquzzaman et Martin Gaedke. “Interoperability in Internet of Things Infrastructure : Classification, Challenges, and Future Work”. en. In : *IoT as a Service*. T. 246. Series Title : Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Cham : Springer International Publishing, 2018, p. 11-18. ISBN : 978-3-030-00409-5 978-3-030-00410-1. DOI : 10.1007/978-3-030-00410-1_2. URL : http://link.springer.com/10.1007/978-3-030-00410-1_2 (visité le 23/05/2023).

[128] Mahda Noura, Mohammed Atiquzzaman et Martin Gaedke. “Interoperability in Internet of Things : Taxonomies and Open Challenges”. en. In : *Mobile Networks and Applications* 24.3 (juin 2019), p. 796-809. issn : 1383-469X, 1572-8153. doi : 10.1007/s11036-018-1089-9. url : http://link.springer.com/10.1007/s11036-018-1089-9 (visité le 10/11/2022).

[129] Natalya F Noy et Mark A Musen. “Anchor-PROMPT : Using Non-Local Context for Semantic Matching”. en. In : *Proc. of the Workshop on Ontologies and Information Sharing at IJCAI-2001* (2001), p. 8.

[130] OMG. *Meta Object Facility (MOF) 2.0 Query View Transformation Specification.pdf*. Avr. 2008. url : https://www.omg.org/spec/QVT/1.0/PDF.

[131] Julio Ottino. “Engineering complex systems”. In : *Nature* 427.6973 (2004), p. 399-399. doi : https://doi.org/10.1038/427399a.

[132] Hervé Panetto et al. “Challenges for the cyber-physical manufacturing enterprises of the future”. en. In : *Annual Reviews in Control* 47 (2019), p. 200-213. issn : 13675788. doi : 10.1016/j.arcontrol.2019.02.002. url : https://linkinghub.elsevier.com/retrieve/pii/S1367578818302086 (visité le 31/08/2022).

[133] Hervé Panetto et al. “New perspectives for the future interoperable enterprise systems”. en. In : *Computers in Industry* 79 (juin 2016), p. 47-63. issn : 01663615. doi : 10.1016/j.compind.2015.08.001. url : https://linkinghub.elsevier.com/retrieve/pii/S0166361515300312 (visité le 13/01/2021).

[134] Cristina Paniagua, Jens Eliasson et Jerker Delsing. “Interoperability Mismatch Challenges in Heterogeneous SOA-based Systems”. en. In : *019 IEEE International Conference on Industrial Technology (ICIT)* (2019), p. 788-193.

[135] Romain Pinquié et al. “An open science platform for benchmarking engineering design researches”. en. In : *Procedia CIRP* 109 (2022), p. 472-477. issn : 22128271. doi : 10.1016/j.procir.2022.05.280. url : https://linkinghub.elsevier.com/retrieve/pii/S2212827122007296 (visité le 22/05/2023).

[136] Milan Pisarić et al. “Towards a Plug-and-Play Architecture in Industry 4.0”. en. In : Novi Sad, Serbia, 2017, p. 136-141.

[137] Michael J. Pratt. “Introduction to ISO 10303—the STEP Standard for Product Data Exchange”. en. In : *Journal of Computing and Information Science in Engineering* 1.1 (mars 2001), p. 102-103. issn : 1530-9827, 1944-7078. doi : 10.1115/1.1354995. url : https://asmedigitalcollection.asme.org/computingengineering/article/1/1/102/445236/Introduction-to-ISO-10303the-STEP-Standard-for (visité le 19/06/2023).

[138] Foster Provost et Tom Fawcett. “Data Science and its Relationship to Big Data and Data-Driven Decision Making”. en. In : *Big Data* 1.1 (mars 2013), p. 51-59. issn : 2167-6461, 2167-647X. doi : 10.1089/big.2013.1508. url : http://www.liebertpub.com/doi/10.1089/big.2013.1508 (visité le 09/06/2023).

[139] Hafizur Rahman et Md. Iftekhar Hussain. “A comprehensive survey on semantic interoperability for Internet of Things : State-of-the-art and research challenges”. en. In : *Transactions on Emerging Telecommunications Technologies* 31.12 (déc. 2020). issn : 2161-3915, 2161-3915. doi : 10.1002/ett.3902. url : https://onlinelibrary.wiley.com/doi/10.1002/ett.3902 (visité le 12/06/2023).

[140] Ragunathan (Raj) Rajkumar et al. “Cyber-physical systems : the next computing revolution”. en. In : *Proceedings of the 47th Design Automation Conference*. Anaheim California : ACM, juin 2010, p. 731-736. isbn : 978-1-4503-0002-5. doi : 10.1145/1837274.1837461. url : https://dl.acm.org/doi/10.1145/1837274.1837461 (visité le 09/06/2023).

[141] Partha Pratim Ray. “A survey of IoT cloud platforms”. en. In : *Future Computing and Informatics Journal* 1.1-2 (déc. 2016), p. 35-46. issn : 23147288. doi : 10.1016/j.fcij.2017.02.001. url : https://linkinghub.elsevier.com/retrieve/pii/S2314728816300149 (visité le 16/10/2020).

[142] Scott A. Renner, Arnon S. Rosenthal et James G. Scarano. “Data Interoperability : Standardization or Mediation.pdf”. In : 1st IEEE metadata conference, avr. 1996.

[143] Leonard Richardson et Sam Ruby. *RESTful web services*. en. OCLC : ocm82671871. Farnham : O’Reilly, 2007. isbn : 978-0-596-52926-0.

[144] Efim Rozenwasser et Rafael Yusupov. *Sensitivity of automatic control systems*. CRC press, 2019.

[145] Hajer Saada et al. “Generation of Operational Transformation Rules from Examples of Model Transformations”. In : *Model Driven Engineering Languages and Systems*. T. 7590. Series Title : Lecture Notes in Computer Science. Berlin, Heidelberg : Springer Berlin Heidelberg, 2012, p. 546-561. isbn : 978-3-642-33665-2 978-3-642-33666-9. doi : 10.1007/978-3-642-33666-9_35. url : http://link.springer.com/10.1007/978-3-642-33666-9_35 (visité le 20/04/2023).

[146] Douglas C Schmidt. “Model-Driven Engineering”. en. In : *IEEE Computer* 39.2 (2006).

[147] Simon Schwichtenberg et al. “Normalizing Heterogeneous Service Description Models with Generated QVT Transformations”. en. In : *Modelling Foundations and Applications*. Sous la dir. de David Hutchison et al. T. 8569. Series Title : Lecture Notes in Computer Science. Cham : Springer International Publishing, 2014, p. 180-195. isbn : 978-3-319-09194-5 978-3-319-09195-2. doi : 10.1007/978-3-319-09195-2_12. url : http://link.springer.com/10.1007/978-3-319-09195-2_12 (visité le 12/07/2021).

[148] S. Sendall et W. Kozaczynski. “Model transformation : the heart and soul of model-driven software development”. en. In : *IEEE Software* 20.5 (sept. 2003), p. 42-45. issn : 0740-7459. doi : 10.1109/MS.2003.1231150. url : http://ieeexplore.ieee.org/document/1231150/ (visité le 03/05/2023).

[149] Weiming Shen et Douglas H. Norrie. “Agent-Based Systems for Intelligent Manufacturing : A State-of-the-Art Survey”. en. In : *Knowledge and Information Systems* 1.2 (mai 1999), p. 129-156. issn : 0219-1377, 0219-3116. doi : 10.1007/BF03325096. url : http://link.springer.com/10.1007/BF03325096 (visité le 22/08/2022).

[150] P. Shvaiko et J. Euzenat. “Ontology Matching : State of the Art and Future Challenges”. en. In : *IEEE Transactions on Knowledge and Data Engineering* 25.1 (jan. 2013), p. 158-176. issn : 1041-4347. doi : 10.1109/TKDE.2011.253. url : http://ieeexplore.ieee.org/document/6104044/ (visité le 25/08/2021).

[151] Delfina SOARES et Luis AMARAL. "Reflections on the Concept of Interoperability in Information Systems :" en. In : *Proceedings of the 16th International Conference on Enterprise Information Systems*. Lisbon, Portugal : SCITEPRESS - Science, 2014, p. 331-339. ISBN : 978-989-758-027-7 978-989-758-028-4 978-989-758-029-1. DOI : 10.5220/0004969703310339. URL : http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0004969703310339 (visité le 23/08/2022).

[152] Richard SOLEY. "Model Driven Architecture". en. In : *OMG white paper* 308.308 (2000), p. 5.

[153] Giorgos STOILOS, Giorgos STAMOU et Stefanos KOLLIAS. "A String Metric for Ontology Alignment". en. In : *The Semantic Web – ISWC 2005*. Sous la dir. de David HUTCHISON et al. T. 3729. Series Title : Lecture Notes in Computer Science. Berlin, Heidelberg : Springer Berlin Heidelberg, 2005, p. 624-637. ISBN : 978-3-540-29754-3 978-3-540-32082-1. DOI : 10.1007/11574620_45. URL : http://link.springer.com/10.1007/11574620_45 (visité le 11/07/2023).

[154] Michael STROMMER, Marion MURZEK et Manuel WIMMER. "Applying Model Transformation By-Example on Business Process Modeling Languages". In : *Advances in Conceptual Modeling – Foundations and Applications*. T. 4802. Series Title : Lecture Notes in Computer Science. Berlin, Heidelberg : Springer Berlin Heidelberg, 2007, p. 116-125. ISBN : 978-3-540-76291-1. DOI : 10.1007/978-3-540-76292-8_14. URL : http://link.springer.com/10.1007/978-3-540-76292-8_14 (visité le 20/04/2023).

[155] Zequn SUN et al. "Bootstrapping Entity Alignment with Knowledge Graph Embedding". en. In : *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. Stockholm, Sweden : International Joint Conferences on Artificial Intelligence Organization, juill. 2018, p. 4396-4402. ISBN : 978-0-9992411-2-7. DOI : 10.24963/ijcai.2018/611. URL : https://www.ijcai.org/proceedings/2018/611 (visité le 06/09/2021).

[156] Zequn SUN et al. "Knowledge Graph Alignment Network with Gated Multi-Hop Neighborhood Aggregation". en. In : *Proceedings of the AAAI Conference on Artificial Intelligence* 34.01 (avr. 2020), p. 222-229. ISSN : 2374-3468, 2159-5399. DOI : 10.1609/aaai.v34i01.5354. URL : https://www.aiide.org/ojs/index.php/AAAI/article/view/5354 (visité le 06/09/2021).

[157] Richard S. SUTTON et Andrew G. BARTO. *Reinforcement learning : An introduction.pdf*. MIT Press. 2018. ISBN : 978-0-262-03924-6.

[158] Gabriele TAENTZER et al. "Model Transformation by Graph Transformation : A Comparative Study". en. In : Montego Bay, Jamaica, 2005, p. 17.

[159] Fei TAO, Jiangfeng CHENG et Qinglin QI. "IIHub : An Industrial Internet-of-Things Hub Toward Smart Manufacturing Based on Cyber-Physical System". In : *IEEE Transactions on Industrial Informatics* 14.5 (2018), p. 2271-2280. DOI : 10.1109/TII.2017.2759178.

[160] Fei TAO et Qinglin QI. "New IT Driven Service-Oriented Smart Manufacturing : Framework and Characteristics". en. In : *IEEE Transactions on Systems, Man, and Cybernetics : Systems* 49.1 (jan. 2019), p. 81-91. ISSN : 2168-2216, 2168-2232. DOI : 10.1109/TSMC.2017.2723764. URL : https://ieeexplore.ieee.org/document/7990538/ (visité le 19/08/2022).

[161] Fei TAO et al. "Data-driven smart manufacturing". en. In : *Journal of Manufacturing Systems* 48 (juill. 2018), p. 157-169. ISSN : 02786125. DOI : 10.1016/j.jmsy.2018.01.006. URL : https://linkinghub.elsevier.com/retrieve/pii/S0278612518300062 (visité le 19/08/2022).

[162] Sobhan Yassipour Tehrani, Steffen Zschaler et Kevin Lano. "Requirements Engineering in Model-Transformation Development : An Interview-Based Study". en. In : *Theory and Practice of Model Transformations*. Sous la dir. de Pieter Van Gorp et Gregor Engels. T. 9765. Series Title : Lecture Notes in Computer Science. Cham : Springer International Publishing, 2016, p. 123-137. isbn : 978-3-319-42063-9 978-3-319-42064-6. doi : 10.1007/978-3-319-42064-6_9. url : http://link.springer.com/10.1007/978-3-319-42064-6_9 (visité le 23/08/2021).

[163] Bayu Distiawan Trisedya, Jianzhong Qi et Rui Zhang. "Entity Alignment between Knowledge Graphs Using Attribute Embeddings". en. In : *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (juill. 2019), p. 297-304. issn : 2374-3468, 2159-5399. doi : 10.1609/aaai.v33i01.3301297. url : https://aaai.org/ojs/index.php/AAAI/article/view/3798 (visité le 22/07/2021).

[164] Zhiying Tu, Gregory Zacharewicz et David Chen. "A federated approach to develop enterprise interoperability". en. In : *Journal of Intelligent Manufacturing* 27.1 (fév. 2016), p. 11-31. issn : 0956-5515, 1572-8145. doi : 10.1007/s10845-013-0868-1. url : http://link.springer.com/10.1007/s10845-013-0868-1 (visité le 23/08/2023).

[165] Žiga Turk. "Interoperability in construction – Mission impossible ?" en. In : *Developments in the Built Environment* 4 (nov. 2020), p. 100018. issn : 26661659. doi : 10.1016/j.dibe.2020.100018. url : https://linkinghub.elsevier.com/retrieve/pii/S2666165920300144 (visité le 23/08/2023).

[166] Gareth Tyson et al. "A Model-Driven Approach to Interoperability and Integration in Systems of Systems". en. In : *7th European Conference on Modelling Foundations and Applications : Workshop on Model-Based Software and Data Integration (MBSDI)* (2011), p. 13.

[167] Saurabh Vaidya, Prashant Ambad et Santosh Bhosle. "Industry 4.0 – A Glimpse". en. In : *Procedia Manufacturing* 20 (2018), p. 233-238. issn : 23519789. doi : 10.1016/j.promfg.2018.02.034. url : https://linkinghub.elsevier.com/retrieve/pii/S2351978918300672 (visité le 09/06/2023).

[168] Martijn Van Otterlo et Marco Wiering. "Reinforcement Learning and Markov Decision Processes". en. In : *Reinforcement Learning*. Sous la dir. de Marco Wiering et Martijn Van Otterlo. T. 12. Series Title : Adaptation, Learning, and Optimization. Berlin, Heidelberg : Springer Berlin Heidelberg, 2012, p. 3-42. isbn : 978-3-642-27644-6 978-3-642-27645-3. doi : 10.1007/978-3-642-27645-3_1. url : http://link.springer.com/10.1007/978-3-642-27645-3_1 (visité le 17/07/2023).

[169] Pal Varga et al. "Making system of systems interoperable – The core components of the arrowhead framework". en. In : *Journal of Network and Computer Applications* 81 (mars 2017), p. 85-95. issn : 10848045. doi : 10.1016/j.jnca.2016.08.028. url : https://linkinghub.elsevier.com/retrieve/pii/S1084804516301965 (visité le 23/08/2023).

[170] Dániel Varró. "Model Transformation by Example". In : *Model Driven Engineering Languages and Systems*. T. 4199. Series Title : Lecture Notes in Computer Science. Berlin, Heidelberg : Springer Berlin Heidelberg, 2006, p. 410-424. isbn : 978-3-540-45772-5 978-3-540-45773-2. doi : 10.1007/11880240_29. url : http://link.springer.com/10.1007/11880240_29 (visité le 20/04/2023).

[171] Petar Veličković et al. "Graph Attention Networks". en. In : *arXiv :1710.10903 [cs, stat]* (fév. 2018). arXiv : 1710.10903. url : http://arxiv.org/abs/1710.10903 (visité le 08/11/2021).

[172] Amanda VENCESLAU et al. "IoT Semantic Interoperability : A Systematic Mapping Study :" en. In : *Proceedings of the 21st International Conference on Enterprise Information Systems*. Heraklion, Crete, Greece : SCITEPRESS - Science et Technology Publications, 2019, p. 535-544. ISBN : 978-989-758-372-8. DOI : 10.5220/0007732605350544. URL : http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0007732605350544 (visité le 12/01/2021).

[173] S. VINOSKI. "It's just a mapping problem". en. In : *IEEE Internet Computing* 7.3 (mai 2003), p. 88-90. ISSN : 1089-7801. DOI : 10.1109/MIC.2003.1200306. URL : http://ieeexplore.ieee.org/document/1200306/ (visité le 19/10/2020).

[174] Tobias WAGNER, Christoph HERRMANN et Sebastian THIEDE. "Industry 4.0 Impacts on Lean Production Systems". en. In : *Procedia CIRP* 63 (2017), p. 125-131. ISSN : 22128271. DOI : 10.1016/j.procir.2017.02.041. URL : https://linkinghub.elsevier.com/retrieve/pii/S2212827117301385 (visité le 08/06/2023).

[175] Jinjiang WANG et al. "Deep learning for smart manufacturing : Methods and applications". en. In : *Journal of Manufacturing Systems* 48 (juill. 2018), p. 144-156. ISSN : 02786125. DOI : 10.1016/j.jmsy.2018.01.003. URL : https://linkinghub.elsevier.com/retrieve/pii/S0278612518300037 (visité le 17/08/2022).

[176] Tiexin WANG, Sebastien TRUPTIL et Frederick BENABEN. "A General Model Transformation Methodology to Serve Enterprise Interoperability Data Sharing Problem". en. In : *Enterprise Interoperability*. Sous la dir. de Marten van SINDEREN et Vincent CHAPURLAT. T. 213. Series Title : Lecture Notes in Business Information Processing. Berlin, Heidelberg : Springer Berlin Heidelberg, 2015, p. 16-29. ISBN : 978-3-662-47156-2 978-3-662-47157-9. DOI : 10.1007/978-3-662-47157-9_2. URL : http://link.springer.com/10.1007/978-3-662-47157-9_2 (visité le 23/08/2021).

[177] Zhichun WANG et al. "Cross-lingual Knowledge Graph Alignment via Graph Convolutional Networks". en. In : *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium : Association for Computational Linguistics, 2018, p. 349-357. DOI : 10.18653/v1/D18-1032. URL : http://aclweb.org/anthology/D18-1032 (visité le 05/10/2021).

[178] Christopher J. C. H. WATKINS et Peter DAYAN. "Q-learning". en. In : *Machine Learning* 8.3-4 (mai 1992), p. 279-292. ISSN : 0885-6125, 1573-0565. DOI : 10.1007/BF00992698. URL : http://link.springer.com/10.1007/BF00992698 (visité le 25/09/2023).

[179] Peter WEGNER. "Interoperability". en. In : *ACM Computing Surveys* 28.1 (1996), p. 285-287. DOI : 10.1145/234313.234424. URL : https://dl.acm.org/doi/10.1145/234313.234424.

[180] Georg WEICHHART, Hervé PANETTO et Arturo MOLINA. "Interoperability in the cyber-physical manufacturing enterprise". en. In : *Annual Reviews in Control* 51 (2021), p. 346-356. ISSN : 13675788. DOI : 10.1016/j.arcontrol.2021.03.006. URL : https://linkinghub.elsevier.com/retrieve/pii/S1367578821000146 (visité le 31/08/2022).

[181] Michael WEISS. "XML Metadata Interchange". In : *Encyclopedia of Database Systems*. Springer US, 2009, p. 3597-3597. ISBN : 978-0-387-39940-9. DOI : 10.1007/978-0-387-39940-9_902. URL : https://doi.org/10.1007/978-0-387-39940-9_902.

[182] M. Wimmer et al. “From the Heterogeneity Jungle to Systematic Benchmarking”. en. In : *Models in Software Engineering*. Sous la dir. de Juergen Dingel et Arnor Solberg. T. 6627. Series Title : Lecture Notes in Computer Science. Berlin, Heidelberg : Springer Berlin Heidelberg, 2011, p. 150-164. isbn : 978-3-642-21209-3 978-3-642-21210-9. doi : 10.1007/978-3-642-21210-9_15. url : http://link.springer.com/10.1007/978-3-642-21210-9_15 (visité le 12/07/2021).

[183] Manuel Wimmer et al. “Towards Model Transformation Generation By-Example”. In : *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*. Proceedings of the 40th Annual Hawaii International Conference on System Sciences. Waikoloa, HI : IEEE, jan. 2007, 285b-285b. isbn : 978-0-7695-2755-0. doi : 10.1109/HICSS.2007.572. url : https://ieeexplore.ieee.org/document/4076959/ (visité le 20/04/2023).

[184] Krzysztof Witkowski. “Internet of Things, Big Data, Industry 4.0 – Innovative Solutions in Logistics and Supply Chains Management”. en. In : *Procedia Engineering* 182 (2017), p. 763-769. issn : 18777058. doi : 10.1016/j.proeng.2017.03.197. url : https://linkinghub.elsevier.com/retrieve/pii/S1877705817313346 (visité le 09/06/2023).

[185] Jifang Wu et al. “DAEOM : A Deep Attentional Embedding Approach for Biomedical Ontology Matching”. en. In : *Applied Sciences* 10.21 (nov. 2020), p. 7909. issn : 2076-3417. doi : 10.3390/app10217909. url : https://www.mdpi.com/2076-3417/10/21/7909 (visité le 11/07/2023).

[186] Yuting Wu et al. *Jointly Learning Entity and Relation Representations for Entity Alignment*. en. arXiv :1909.09317 [cs]. Sept. 2019. url : http://arxiv.org/abs/1909.09317 (visité le 06/07/2023).

[187] Yuejia Xiang et al. *OntoEA : Ontology-guided Entity Alignment via Joint Knowledge Graph Embedding*. en. arXiv :2105.07688 [cs]. Mai 2021. url : http://arxiv.org/abs/2105.07688 (visité le 11/07/2023).

[188] Yuchen Yan et al. “Dynamic Knowledge Graph Alignment”. en. In : t. 35. 2021, p. 4564-4572. doi : https://doi.org/10.1609/aaai.v35i5.16585.

[189] Jiaxuan You et al. “Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation”. en. In : t. 31. 2018. isbn : 978-1-5108-8447-2.

[190] Li Yujian et Liu Bo. “A Normalized Levenshtein Distance Metric”. In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29 (avr. 2007), p. 1091-195. doi : 10.1109/TPAMI.2007.1078. url : https://ieeexplore.ieee.org/abstract/document/4160958.

[191] Gregory Zacharewicz et al. “Model Driven Interoperability for System Engineering”. en. In : *Modelling* 1.2 (oct. 2020), p. 94-121. issn : 2673-3951. doi : 10.3390/modelling1020007. url : https://www.mdpi.com/2673-3951/1/2/7 (visité le 24/04/2023).

[192] Marcia Lei Zeng. “Interoperability”. In : *KO Knowledge Organization* 46.2 (2019), p. 122-146. doi : doi.org/10.5771/0943-7444-2019-2-122.

[193] Weixin Zeng et al. “Reinforcement Learning based Collective Entity Alignment with Adaptive Features”. en. In : *arXiv :2101.01353 [cs]* (jan. 2021). arXiv : 2101.01353. url : http://arxiv.org/abs/2101.01353 (visité le 03/09/2021).

[194] Frank Zenker et Peter Gärdenfors, éd. *Applications of Conceptual Spaces*. en. Cham : Springer International Publishing, 2015. isbn : 978-3-319-15020-8 978-3-319-15021-5. doi : 10.1007/978-3-319-15021-5. url : http://link.springer.com/10.1007/978-3-319-15021-5 (visité le 19/08/2021).

[195] Qingheng Zhang et al. "Multi-view Knowledge Graph Embedding for Entity Alignment". en. In : *arXiv :1906.02390 [cs]* (juin 2019). arXiv : 1906.02390. url : http://arxiv.org/abs/1906.02390 (visité le 23/07/2021).

[196] Xiang Zhao et al. "An Experimental Study of State-of-the-Art Entity Alignment Approaches". en. In : *IEEE Transactions on Knowledge and Data Engineering* (2020), p. 1-1. issn : 1041-4347, 1558-2191, 2326-3865. doi : 10.1109/TKDE.2020.3018741. url : https://ieeexplore.ieee.org/document/9174835/ (visité le 06/07/2023).

[197] Pai Zheng et al. "Smart manufacturing systems for Industry 4.0 : Conceptual framework, scenarios, and future perspectives". en. In : *Frontiers of Mechanical Engineering* 13.2 (juin 2018), p. 137-150. issn : 2095-0233, 2095-0241. doi : 10.1007/s11465-018-0499-5. url : http://link.springer.com/10.1007/s11465-018-0499-5 (visité le 04/10/2023).

[198] Ray Y. Zhong et al. "Intelligent Manufacturing in the Context of Industry 4.0 : A Review". en. In : *Engineering* 3.5 (oct. 2017), p. 616-630. issn : 20958099. doi : 10.1016/J.ENG.2017.05.015. url : https://linkinghub.elsevier.com/retrieve/pii/S2095809917307130 (visité le 18/08/2022).

Annexe A

Classes aplaties de la transformation families2persons

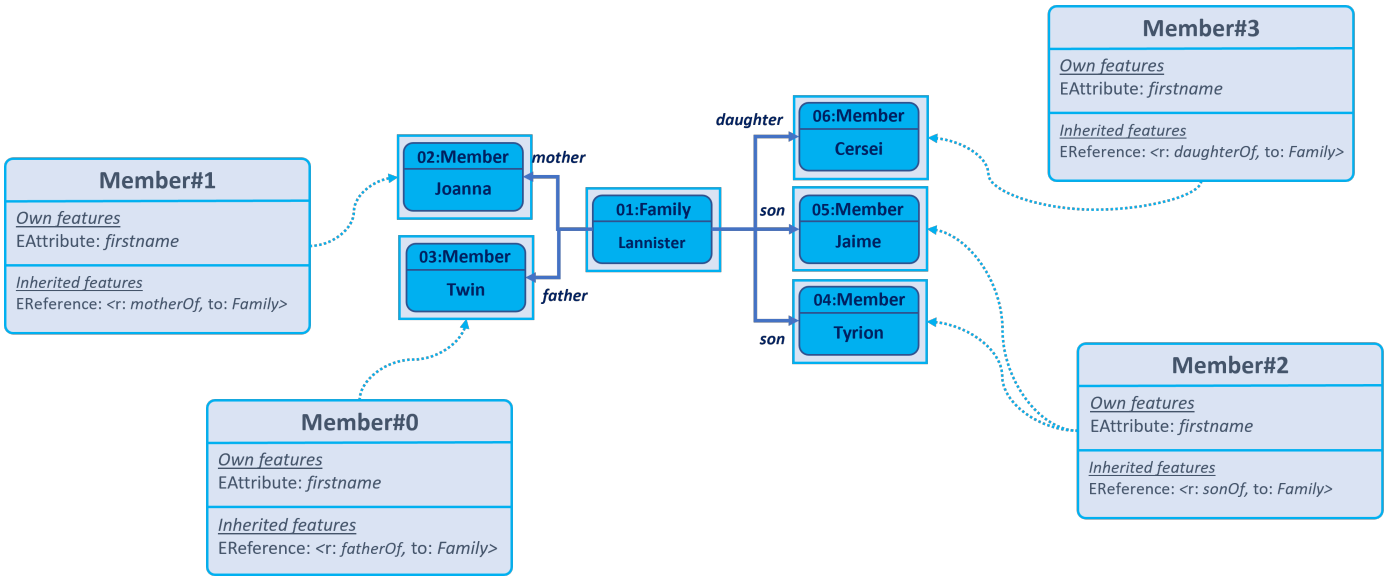


FIGURE A.1 – Classes aplaties du métamodèle *Families*

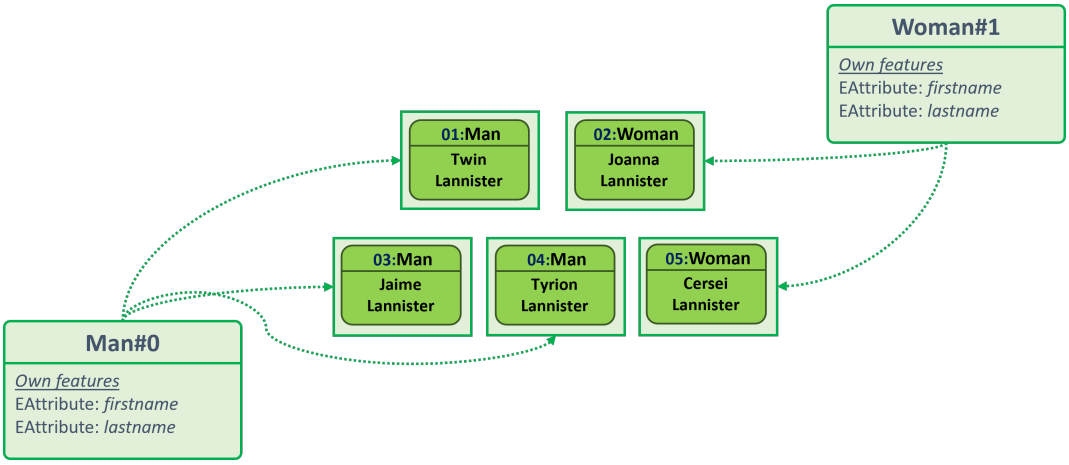


FIGURE A.2 – Classes aplaties du métamodèle *Persons*

Annexe B

Cas d'étude n°1 : Automatisation de l'édition des document *As-Built*

B.1 Données d'entraînement : transformation prog66 vers AsBuilt

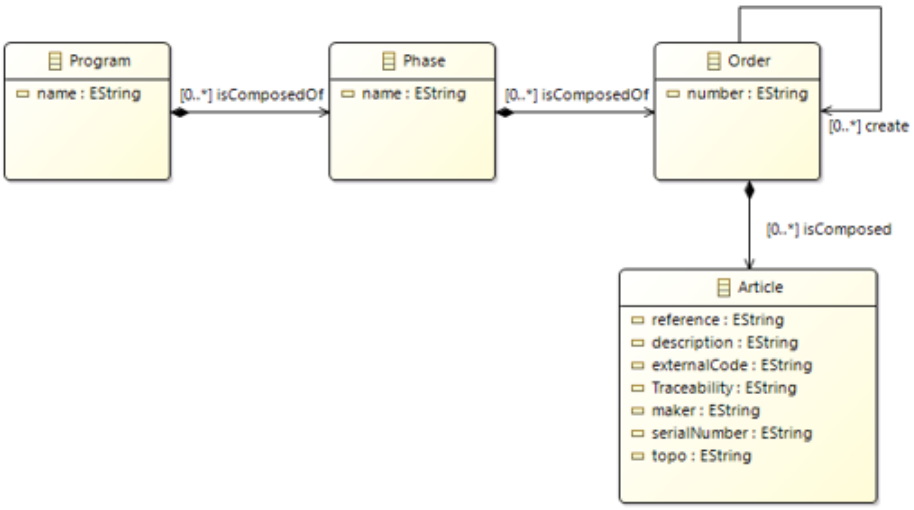


FIGURE B.1 – Diagramme de classes du métamodèle *prog66*

```
<?xml version="1.0" encoding="UTF-8"?>
<Prog66:Program
    xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:Prog66="Prog66"
    xsi:schemaLocation="Prog66 Prog66.ecore"
    name="SAT">
  <isComposedOf
      name="SMMOD">
    <isComposedOf
        number="251341"
        create="//@isComposedOf.0/@isComposedOf.2">
      <isComposed
          reference="97537100"
          description="GYRO REGYS20"
          externalCode="2356.11.00.000.9 SAGEM"
          Traceability="000010000000028"
          maker="F1"
          serialNumber="28"
          topo="GYRO"/>
    </isComposedOf>
    <isComposedOf
        number="272461"
        create="//@isComposedOf.0/@isComposedOf.2">
      <isComposed
          reference="97537100"
          description="GYRO REGYS20"
          externalCode="2356.11.00.000.9 SAGEM"
          Traceability="000010000000029"
          maker="F1"
          serialNumber="29"
          topo="GYRO"/>
    </isComposedOf>
    <isComposedOf
        number="271356">
      <isComposed
          reference="120681400"
          description="GYRO REGYS20 EQUIPE"
          externalCode="4CNREGYSP40A"
          Traceability="000000000000020"
          topo="GYRO1"/>
      <isComposed
          reference="120681400"
          description="GYRO REGYS20 EQUIPE"
          externalCode="4CNREGYSP40A"
          Traceability="000000000000021"
          topo="GYRO2"/>
    </isComposedOf>
  </isComposedOf>
</Prog66:Program>
```

FIGURE B.2 – Modèle *prog66.xmi* (modèle source)

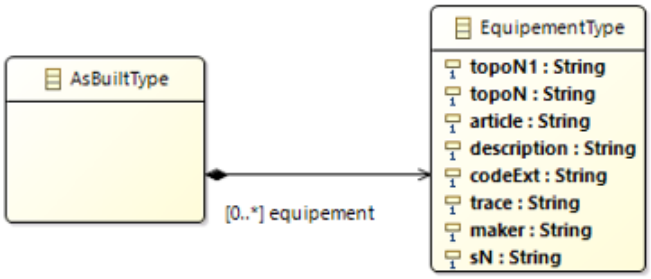


FIGURE B.3 – Diagramme de classes du métamodèle *AsBuilt*

```
<?xml version="1.0" encoding="ASCII"?>
<AsBuilt:TableType
    xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI"
    xmlns:AsBuilt="file:/c:/Users/Quentin/Dev/microservices/Services/views/AsBuilt_package/xsd/AsBuilt.xsd">
  <row
    topoN1="GYRO1"
    topoN="GYRO"
    article="97537100"
    description="GYRO REGYS20"
    codeExt="2356.11.00.000.9 SAGEM"
    trace="000010000000029"
    maker="F1"
    sN="29"
    disparityAsBuilt="-"/>
  <row
    topoN1="GYRO2"
    topoN="GYRO"
    article="97537100"
    description="GYRO REGYS20"
    codeExt="2356.11.00.000.9 SAGEM"
    trace="000010000000028"
    maker="F1"
    sN="28"
    disparityAsBuilt="-"/>
</AsBuilt:TableType>
```

FIGURE B.4 – Modèle *AsBuilt.xmi* (modèle cible)

Arts et Métiers Sciences et Technologies | Quentin BRILHAULT | HESAM UNIVERSITÉ

Résumé : La transformation numérique de l'industrie oblige les entreprises à améliorer la continuité numérique au sein de leur organisation en renforçant la capacité de leurs systèmes distribués à communiquer et à coordonner leurs activités, afin d'assurer des tâches complexes, telles que la planification intelligente, la détection d'anomalies, ou encore, la réduction de l'empreinte écologique des processus industriels. Dans un environnement numérique instable et fortement hétérogène, en constante évolution tant sur le plan technique avec l'émergence de nouvelles technologies que sur le plan organisationnel avec les exigences accrues en matière d'agilité et d'adaptabilité des chaînes de production, l'établissement et le maintien de l'interopérabilité entre les systèmes est une tâche cruciale. Les transformations de modèles, pierre angulaire de l'Architecture Dirigée par les Modèles, pourraient apporter une solution concrète aux exigences d'une intéropérabilité dynamique et durable entre les systèmes. D'autant plus qu'un nouveau paradigme, utilisant des techniques d'apprentissage automatique, pourrait simplifier la création et la maintenance des modèles de transformation en apprenant automatiquement les règles de transformation entre les modèles. Deux contributions sont présentées dans de ce manuscrit : (1) Une application des principes de l'apprentissage par renforcement pour dériver automatiquement les règles de transformation ; (2) Un protocole expérimental pour évaluer et valider la capacité de l'approche proposée à automatiquement inférer des modèles de transformation tout en respectant les spécifications de l'Industrie 4.0. Un *benchmark* des approches existantes conclut ce travail, montrant l'efficacité des techniques d'apprentissage par renforcement dans l'apprentissage des règles de transformation qui relient deux métamodèles différents.

Mots clés : Continuité numérique, Interopérabilité des systèmes, Transformation des modèles, Architecture Dirigée par les Modèles, Apprentissage par renforcement.

Abstract : The digital transformation of industry is forcing companies to improve digital continuity within their organization by enhancing the ability of their distributed systems to communicate and coordinate their activities to ensure complex tasks, such as smart planning, anomaly detection, and ecological footprint reduction. In an unstable and highly heterogeneous digital environment that is constantly evolving both technically with the emergence of new technologies and organizationally with the increased demands for agility and adaptation of production chains, establishing and maintaining interoperability between systems is a crucial task. Model transformations (MT), the cornerstone of Model-Driven Architecture (MDA), could provide a concrete solution to the requirements of dynamic and sustainable interoperability between systems. A new paradigm using machine learning techniques could simplify the creation and maintenance of MT by automatically learning transformation rules between models. The two contributions in this manuscript are : (1) An application of the *Reinforcement Learning* (RL) principles, in particular *Q-learning*, is used to automatically derive transformation rules and learn reusable MT as interoperability functions between metamodels ; (2) An experimental protocol to evaluate the ability of approaches to automatically construct MT and meet the requirements of Industry 4.0. A benchmark of existing approaches concludes this work, showing the effectiveness of RL techniques in learning the transformation rules that connect two domain modelling languages.

Keywords : Digital continuity, System interoperability, Model transformation, Model-driven architecture, Reinforcement learning.