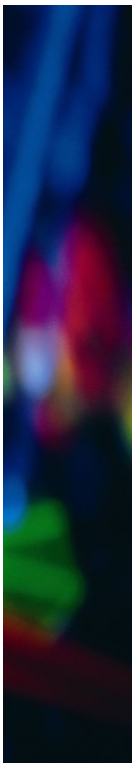
Utilisation de l'ingénierie dirigée par les modèles pour l'agrégation continue de données hétérogènes : application à la supervision de réseaux de gaz

Ahmed Ahmed



2018-ENAM-0049

École doctorale n° 432 : Science des Métiers de l'ingénieur

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

l'École Nationale Supérieure d'Arts et Métiers

Spécialité " Informatique"

présentée et soutenue publiquement par

Ahmed AHMED

17 décembre 2018

Utilisation de l'ingénierie dirigée par les modèles pour l'agrégation continue de données hétérogènes :

Application à la supervision de réseaux de gaz

Directeur de thèse : **Lionel ROUCOULES**
Co-encadrement de la thèse : **Mathias KLEINER**

Jury

M. Jean-Philippe BABAU, Professeur, Lab-STICC, Université de Bretagne Occidentale	Président
M. Vincent CHEUTET, Professeur, DISP, INSA Lyon	Rapporteur
M. Gerson SUNYE, Maître de conférences, LS2N, Université de Nantes	Rapporteur
M. Jacques MEZHRAHID, Ingénieur, Sogeti-HT Capgemini	Examinateur
M. Lionel ROUCOULES, Professeur, LISPEN, Arts et Métiers ParisTech	Examinateur
M. Mathias KLEINER, Maître de conférences, LISPEN, Arts et Métiers ParisTech	Examinateur

T H È S E

Arts et Métiers ParisTech - Campus d'Aix-en-Provence
Laboratoire d'Ingénierie des Systèmes Physiques et Numériques (LISPEN EA 7515)

Remerciements

Les travaux présentés dans cette thèse ont fait l'objet d'une convention CIFRE entre la société Sogeti-HT - Capgemini et le Laboratoire d'Ingénierie des Systèmes Physiques et Numériques (LISPEN) aux Arts et Métiers ParisTech d'Aix-en-Provence.

Cette thèse n'aurait pas été possible sans l'intervention, consciente, d'un grand nombre de personnes. Je reconnais que chacune a apporté, à des degrés divers, mais avec une égale bienveillance, une contribution positive à la finalisation de cette thèse. Je tiens à remercier ces personnes d'avoir fait de ces quatre années de thèse un souvenir inoubliable.

Je tiens à remercier en premier lieu mon directeur de thèse M. Lionel ROUCOULES, Professeur des Universités aux Arts et Métiers ParisTech d'Aix-en-Provence, ainsi que mon co-encadrant, M. Mathias KLEINER, Maître de Conférences aux Arts et Métiers ParisTech d'Aix-en-Provence. J'ai eu l'opportunité d'être suivi par ces deux personnes exceptionnelles. Merci à vous deux de m'avoir encadré et plus important encore de m'avoir soutenu tout au long de cette thèse. Leurs connaissances, leurs critiques, et leurs conseils constructifs m'ont permis de mener à bien ce travail. Merci pour votre grand effort dans la correction de ce manuscrit.

Je tiens aussi à remercier l'ensemble de mes co-encadrants techniques et administratifs à Sogeti-HT -Capgemini. Je tiens à remercier particulièrement M. Rémy GAUDY sans qui cette thèse CIFRE n'aurait surement jamais vu le jour. Son aide si précieuse, sa présence et son soutien constant m'ont été source de motivation, de réconfort et de persévérance. J'ai une pensée particulière à M. Bertrand LARAT, M. Eric Oursel, M. Etienne ROSSIGNON, M. Georges LUBRANO, M. Christophe HUGOT, et à toutes les personnes de la société avec qui j'ai travaillé.

J'adresse tous mes remerciements à M. Vincent CHEUTET, Professeur des Universités à INSA Lyon, ainsi qu'à M. Gerson SUNYE, Maître de Conférences à l'Université de Nantes, de l'honneur qu'ils m'ont fait en acceptant d'être rapporteurs de cette thèse. Merci pour leurs remarques constructives, vos questions, et vos idées qui m'ont permis d'améliorer la qualité de mon manuscrit. Je suis reconnaissante à M. Jacques MEZHRAHID, Ingénieur à Sogeti-HT Capgemini, d'examiner les travaux présentés dans ce mémoire. Je tiens à remercier également M. Jean-Philippe BABAU, Professeur à l'Université de Brest, pour avoir accepté de présider le jury de thèse et d'avoir apporté sa caution scientifique à mon travail.

Je remercie très chaleureusement l'équipe d'ingénieurs du laboratoire LISPEN. Je tiens à remercier particulièrement M. Laurent VALLET pour l'aide précieuse qu'il m'a apportée tout au long de ce travail. Je remercie chaleureusement tous ceux que j'ai pu côtoyer ces dernières années à l'ENSAM d'Aix-en-Provence. J'exprime ma gratitude à Madame CAUQUIL Grazyna, Assistante LISPEN, qui était toujours disponible pour m'aider dans les démarches administratives. Je voudrais aussi remercier mes camarades thésards pour tous les bons moments passés ensemble. Merci pour vos aides précieuses, vos disponibilités de tout instant, vos conseils et vos bonnes humeurs quotidiennes. Je vous souhaite le meilleur pour vos carrières et vos vies personnelles. J'ai une pensée particulière à Widad ES-SOUFI pour avoir toujours été présente et ton écoute. Un grand merci aussi pour ton aide lors de la rédaction des lettres administratives en langue française, pour ton soutien et tes encouragements toujours au bon moment. Je me souviendrai toujours de tes blagues et ton sens de l'humour. Je te souhaite beaucoup de bonheur pour la suite.

Un grand merci aussi à mes amis pour leur soutien. Ils étaient toujours à mes côtés pendant les moments heureux et difficiles pour me pousser et me motiver.

Enfin, les mots les plus simples étant les plus forts, j'adresse toute mon affection à ma famille qui m'a soutenue et encouragée : à mon père, ma mère, ma soeur, mon beau-frère. Malgré la distance, leur confiance, leur encouragement, leur tendresse, leur amour me portent et me guident tous les jours. Merci d'avoir fait de moi ce que je suis aujourd'hui. Je tiens à remercier une personne très spéciale, ma femme Arzaq, pour sa gentillesse, ses sacrifices et ses encouragements pendant la rédaction de cette thèse. Ta présence à mes côtés m'a permis de dépasser toutes les difficultés rencontrées sur mon chemin. J'ai tellement de chance d'avoir une si belle famille.

Ahmed AHMED

Table des matières

Liste des figures

Liste des tableaux

Introduction

Durant les dix dernières années, la révolution numérique et l'essor soudain des nouvelles technologies ont fortement impacté le monde dans lequel nous vivons et travaillons. L'infrastructure des technologies de l'information et de la communication dans l'environnement industriel a vécu des changements qui contribuent à son évolution. Cela inclut les évolutions des PC, des systèmes complexes, des réseaux, du cloud computing et des services, des données massives, de l'intelligence artificielle, des capteurs intelligents, etc.

Aujourd'hui, les objets, les gens, les services peuvent être connectés grâce à l'internet. Par exemple, vous pouvez régler à distance la température de votre maison, contrôler la lumière et l'appareil électrique par téléphone, etc. Dans un contexte plus large comme le réseau d'électricité intelligent, si le prix d'électricité augmente pendant la période de pointe, le système intelligent de chauffage de maison doit réduire la puissance afin de diminuer la consommation.

En effet, l'infrastructure informatique et l'infrastructure industrielle évoluent de manière à passer des systèmes monolithiques à des systèmes et des équipements hétérogènes, autonomes et largement distribués [Hasselbring, 2000]. L'hétérogénéité se produit en raison de la différence des langages de programmation, des matériels hardware, des plate-formes d'exploitations, des formats de données différents, des différents mécanismes de communication, etc. Tous les systèmes ne peuvent pas coexister de manière isolée, mais exigent que leurs données soient partagées de manière à accroître la productivité de l'entreprise. En fait, nous progressons vers des systèmes complexes plus vastes où des millions d'appareils doivent être intégrés tel que dans les environnements intelligents émergents : Smart Grids, Smart Gas Network, Smart Cities, Smart Mobility, Street Lighting, Home Automation, Future Industry.

Ce travail se déroule dans le contexte de réseau de distribution de gaz français. Dans cet environnement, il y a plusieurs systèmes hétérogènes et distribués, quelques uns sont des systèmes existants (Legacy systems) et quelques autres sont de nouveaux systèmes. Tous ces systèmes doivent être capables de collaborer et d'échanger les données entre eux. Alors que nous aboutissons vers un grand système complexe, l'exigence d'une solution d'interopérabilité peu coûteuse et rapide devient un besoin essentiel [He and Xu, 2014].

Les solutions utilisent aujourd'hui des codages en dur (Hard coded solutions) pour réaliser de mapping entre les systèmes. D'autres solutions imposent une norme ou une technologie à suivre par les systèmes afin de favoriser l'interopérabilité entre eux. Comme dans la réalité où les personnes hétérogènes ne peuvent pas se conformer à une seule langue, les systèmes d'information ne peuvent pas non plus se conformer à une seule norme ou technologie. Ainsi, ce travail propose l'étude et le développement d'une architecture d'interopérabilité **générique**, **modulaire**, **agnostique** et **extensible** basée sur des principes de l'architecture dirigée par les modèles et les concepts de la séparation de préoccupations. Il vise à promouvoir l'interopérabilité et l'échange de données entre les systèmes hétérogènes en temps réels sans obliger les systèmes à se conformer à des normes ou technologies spécifiques.

Structure du manuscrit

La suite du manuscrit se divise en 6 chapitres distribués entre trois grandes parties I) la problématique, II) la proposition, et III) le cas d'étude, la validation et la conclusion dont le contenu de chacun est synthétisé ci-après.

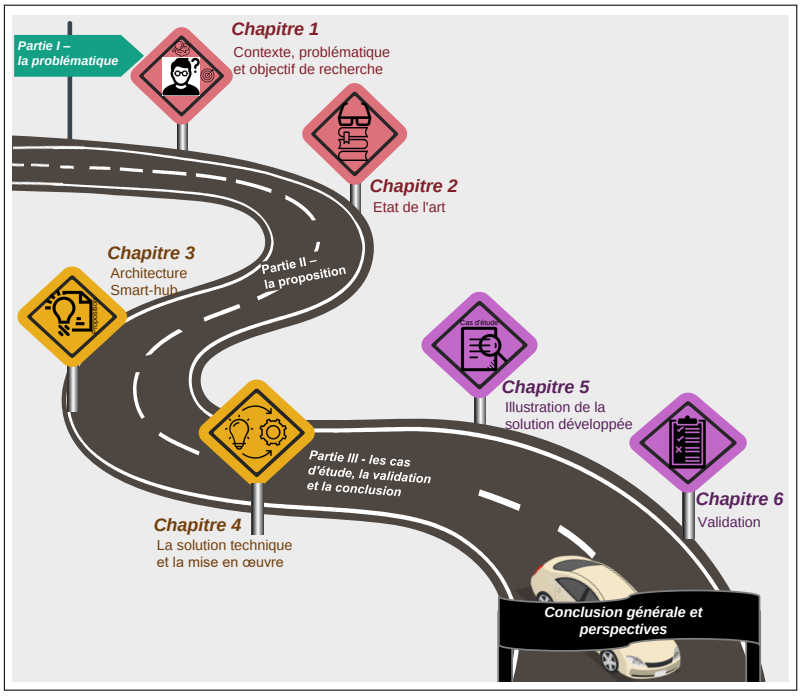


FIGURE 1 – Feuille de route du manuscrit

- **Partie I - la problématique :**
  - Le **chapitre 1** a pour objectif d'introduire le contexte global et objectif de nos travaux de thèse. D'abord, il présente un aperçu rapide de l'histoire de la révolution industrielle. Ensuite, il discute du problème d'interopérabilité auquel nous sommes confrontés aujourd'hui. Ce chapitre présente également les causes de ce problème. Il finit en introduisant l'objectif, la contribution et le contexte industriel de notre étude.
  - Le **chapitre 2** est un état de l'art qui synthétise les grandes approches qui pourraient résoudre le problème de l'interopérabilité pour échanger agréger les données entre les systèmes hétérogènes. L'état de l'art est abordé selon trois axes afin de répondre à nos fonctions. Le premier axe expose les solutions pour gérer les niveaux d'interopérabilité séparément. Le deuxième axe présente des solutions plus génériques en prenant en compte tous les niveaux d'interopérabilité à la fois. Il compare aussi les solutions d'interopérabilité existantes en fonction de certains critères. Le troisième axe introduit l'approche de l'ingénierie dirigée par les modèles que nous utiliserons pour proposer une solution afin d'atteindre nos objectifs.
- **Partie II - la proposition :**
  - Le **chapitre 3** est la conception de l'architecture que nous proposons dans cette thèse. Il est abordé selon trois sections. La première section introduit un exemple de contexte simplifié afin de faciliter la discussion et la présentation de la proposition. La deuxième section introduit l'analyse fonctionnelle externe de l'architecture proposée. Ensuite, la troisième section présente l'analyse fonctionnelle interne de l'architecture proposée.
  - Le **chapitre 4** correspond à la solution technique et concrète de l'architecture que nous proposons dans cette thèse. D'abord, il présente les outils et les techniques tels qu'EMF, ATL, etc. qui sont utilisés pour mettre en œuvre la proposition. Ensuite, il présente la réalisation du prototypage de l'architecture proposé.
- **Partie III - les cas d'étude, la validation et la conclusion :**
  - Le **chapitre 5** montre l'application de notre architecture proposée sur un cas d'étude simplifié puis un cas d'usage de dimension réelle dans le domaine de supervision de réseaux de gaz.
  - Le **chapitre 6** est une validation de notre proposition pour vérifier si notre solution accomplit son but prévu en répondant aux objectifs définis dans le premier chapitre.

– Enfin, une conclusion résume les apports de notre thèse au domaine de l'interopérabilité afin d'agréger et d'échanger des données entre des systèmes hétérogènes. Nous terminerons ce mémoire en proposant quelques perspectives à ce travail.

Publications

JOURNAUX INTERNATIONAUX

1. A. Ahmed, M. Kleiner and L. Roucoules, "Model-Based Interoperability IoT Hub for the Supervision of Smart Gas Distribution Networks", *In IEEE Systems Journal*, vol. , no. , pp. 1-8, 2018. ISSN 1932-8184. doi : 10.1109/JSYST.2018.2851663 (in press).

CONFÉRENCES INTERNATIONALES

2. A. Ahmed, L. Roucoules, R. Gaudy, and B. Larat. *Data aggregation architecture "Smart-hub" for heterogeneous systems in industrial environment*, pages 851–859. Springer International Publishing, Cham, 2017. ISBN 978-3-319-45781-9. doi : 10.1007/978-3-319-45781-9_-85.

3. A. Ahmed, M. Kleiner, L. Roucoules, R. Gaudy and B. Larat, "Model-based interoperability solutions for the supervision of smart gas distribution networks", *In 2016 11th System of Systems Engineering Conference (SoSE)*, pp. 1-5, 2016.

Première partie

Problématique

Chapitre 1

Contexte, problématique et objectif de recherche

Sommaire

Au cours des récentes avancées en matière d'interopérabilité, le besoin d'un environnement interopérable entre les systèmes est devenu inévitable en raison de l'hétérogénéité entre les systèmes d'information. L'hétérogénéité se produit en raison de la différence des langages de programmation, des matériels hardware, des plate-formes d'exploitations, des formats de données différents, etc. Même si chaque système peut fonctionner de façon autonome un moyen de collaboration et d'échange de données doit être un besoin essentiel entre ces systèmes afin de promouvoir l'interopérabilité entre eux. D'une manière générale, l'interopérabilité est simplement la capacité d'échanger et d'utiliser les informations.

Le but de ce chapitre est de présenter le contexte, les problématiques et les objectifs de ce travail de recherche. Avant d'exposer notre proposition, il est indispensable de faire un tour historique sur les révolutions industrielles dans section 1.1. Ensuite, les sections 1.2 et 1.3 présentent l'importance et le rôle de l'interopérabilité suivie par les causes qui conduisent au problème d'interopérabilité. Puis, nous présentons notre question de recherche, notre but, nos objectifs et nos contributions autour de l'interopérabilité dans les sections 1.4, 1.5 et 1.6.

Ce sont les aspects essentiels qui définissent nos exigences. Enfin, la section 1.7 présente le contexte qui a donné lieu à l'apparition de la problématique ainsi que le cas d'étude dans lequel notre travail sera testé et appliqué.

1.1 La révolution industrielle

La révolution industrielle est un concept qui a fondamentalement changé notre société et notre économie [Bloem et al., 2014]. Elle a évolué à travers un certain nombre de phases comme illustré dans la figure 1.1. La première révolution industrielle apparaît à la fin du XVIIIe. Elle est liée à l'utilisation de la machine à vapeur comme moteur pour actionner des machines. Ensuite, depuis la fin du XIXe siècle, la dépendance à l'utilisation de nouvelles sources d'énergie comme l'électricité, le gaz, et le pétrole pour les productions massives conduit à la deuxième révolution industrielle. La troisième révolution industrielle amorce dans le dernier tiers du XXe siècle. Elle se caractérise principalement par l'utilisation de l'électronique et de l'informatique, qui rendent possibles la production de matériels miniaturisés et l'automatisation poussée de la production. Ces systèmes informatiques et les logiciels, qui sont utilisés pour gérer les opérations industrielles, sont référencés par le terme Technologie opérationnelle (Operational Technology - OT en anglais). Cela inclut les systèmes de contrôle industriel (Industrial Control Systems - ICS en anglais), les systèmes cyber-physiques (CPS), etc. Autrement dit, il reflète l'échange vertical de données entre les dispositifs et les logiciels. Ce terme se pose pour distinguer les systèmes OT des systèmes traditionnels qui sont référés par le terme la technologie de l'information (Information Technology - IT) en anglais). Le terme IT reflète l'échange horizontal de données entre les logiciels de l'entreprise.

L'essor soudain des nouvelles technologies a fortement impacté le fonctionnement des industries : l'amélioration continue des différents processus et de l'ensemble des opérations de production ou de maintenance a été rendue réalisable notamment grâce à l'exploitation massive et intensive des moyens numériques. Ce résultat sur la naissance de la quatrième vague de la révolution industrielle qui se caractérise par une interconnexion des machines et des systèmes.

Aujourd'hui, l'internet est devenu une partie de notre vie et il permet de connecter les gens, les objets, les services. L'internet des Objets (IdO), Internet of Things (IoT) en anglais, est la connectivité de l'équipement généralement appelé « objets » par des protocoles de communication de l'internet. Ces objets, comme le RFID, les capteurs, les objets connectés, les appareils portables, les systèmes embarqués, les actionneurs, etc. coopèrent et communiquent entre eux pour atteindre un objectif commun. L'IdO vise à proposer des solutions prometteuses pour transformer l'opération et le rôle de plusieurs systèmes existants dans les systèmes industriels comme les systèmes logistiques, les systèmes de fabrications, les systèmes de transport, réseaux de gaz, etc. Par exemple, si le compteur intelligent (IdO) dans le

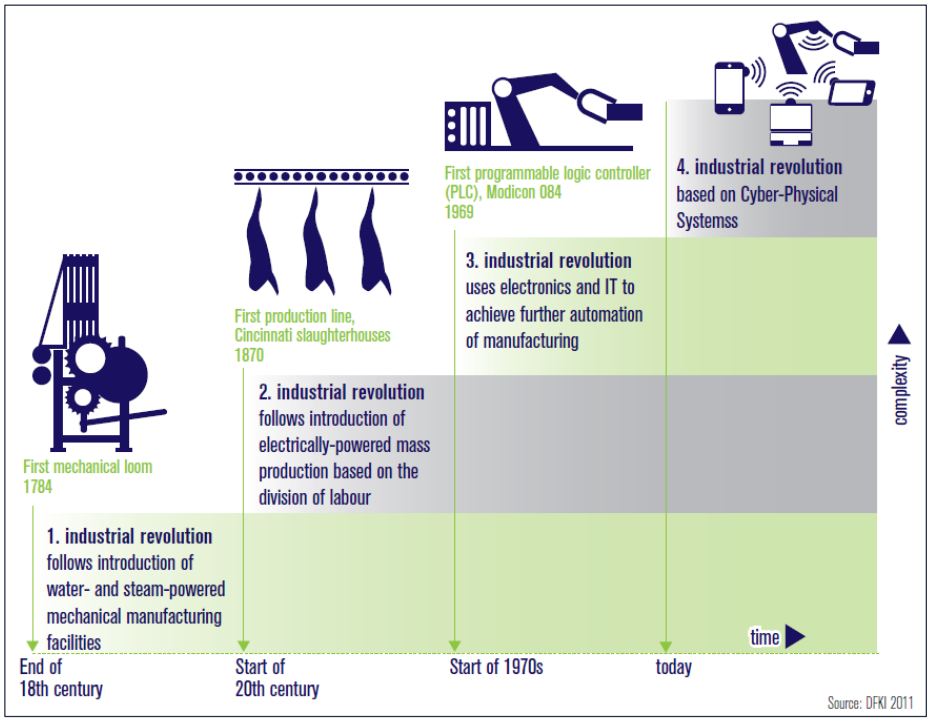


FIGURE 1.1 – Les étapes de la révolution industrielle [Henning Kagermann, 2013]

réseau de gaz détecte une augmentation du prix de l'électricité pendant la haute saison, le compteur intelligent communique avec d'autres appareils (IdO) pour réduire leur puissance afin de réduire la consommation d'énergie.

Le nombre d'équipements ou d'objets qui sont reliés au réseau augmente de façon exponentielle. Selon l'analyse de CISCO, 50 milliards d'appareils seront connectés à l'internet [Evans, 2011]. En plus, le marché IdO valait 14.4 billions de dollars en 2022 [Bradley et al., 2013]. Ils sont appliqués à de nombreux domaines tels que la logistique, la santé, l'énergie et l'industrie, etc. [CEN-CENELEC-ETSI and Coordination, 2012, Drath and Horch, 2014, Su et al., 2011]. L'internet des Objets (IdO) est l'outil clé et l'une des caractéristiques qui a donnée naissance à quatrième révolution industrielle.

La dernière et la quatrième révolution industrielle "Industrie 4.0" a été introduite en 2011 par le gouvernement fédéral allemand comme une stratégie de haute technologie pour 2020 qui encourage la révolution numérique des industries [Henning Kagermann, 2013]. C'est un phénomène dans le but de transformer la fabrication et d'autres processus industriels partout dans le monde afin d'augmenter la productivité et de diminuer le coût.

Il y a plusieurs définitions pour l'industrie 4.0 données par différentes associations, organisations, institutions de recherche et industries. Par exemple, l'association des entreprises de télécommunications allemandes BITKOM relève pas moins de 104 définitions, caractérisations et descriptions différentes dans son document de présentation de l'industrie 4.0 [Bidet-Mayer and Ciet, 2016]. Donc, il est devenu difficile de converger vers une définition

commune. Nous présentons ici quelques définitions. Selon [Schumacher et al., 2016], l'Industrie 4.0 se réfère aux récentes avancées technologiques où l'Internet et les technologies de soutien (par exemple les systèmes embarqués) servent de colonne vertébrale à l'intégration des objets physiques, les acteurs humains, des machines intelligentes, des lignes et des processus produits à travers les frontières organisationnelles pour former un nouveau type d'intelligence, chaîne de valeur agile et en réseau. Trappey et al. [2017] définit l'Industrie 4.0 comme un concept général permettant la fabrication avec les éléments de l'intelligence tactique en utilisant des techniques et des technologies telles que l'Internet des objets, le cloud computing et le big data. [Moeuf et al., 2017] définit l'Industrie 4.0 comme étant l'utilisation de nouvelles technologies et techniques (capteurs, cloud, analyse de données massives, etc.) pour une plus grande communication entre les différents objets et/ou ressources de l'entreprise (personnes, machines, produits, clients, etc.), dans le but d'être connectés en temps réel sur l'ensemble de ses ressources. Nous avons limité la portée de notre revue à cette dernière définition de l'industrie 4.0 donnée par [Moeuf et al., 2017]. En effet, cette définition est la plus claire, car elle reflète le contexte de notre travail qui se concentre sur la connexion de systèmes hétérogènes en temps réel.

L'industrie 4.0 annonce une toute nouvelle façon d'informatiser nos entreprises où toutes les machines, les logiciels et les objets, et les humains communiquent et coopèrent entre eux pour prendre des décisions importantes qui ont un impact sur l'entreprise. À l'origine, le terme Industrie 4.0 n'était utilisé que pour la fabrication, mais il s'est répandu dans divers domaines tels que les gaz intelligents, les bâtiments intelligents, etc. Dans l'écriture anglophone, l'industrie 4.0 est référencée par Smart Factory, Smart Industry, Factory of the Future, Industry of the Future, Digital Factory, etc. Un composant important de l'industrie 4.0 est le système cyber-physique (CPS). CPS est la fusion du monde physique et le monde virtuel. Il étend les capacités du monde physique par calcul où les systèmes de calcul supervisent et contrôlent les processus physiques [Lee, 2008]. L'intégration de l'IdO et CPS ensemble donne lieu aux usines intelligentes où les êtres humains, les objets (machines) et les ressources (données) communiquent les uns avec les autres aussi naturellement que dans un réseau social pour faire des actions selon le contexte actuel (cf. figure 1.2 [1]). Les capacités des nouvelles technologies à se connecter à tout et d'échanger sur différentes plate-formes sur différents formats sont possibles si et seulement si tous les participants à les échanges d'informations sont interopérables.

[1] https ://kr.fotolia.com/id/63778652

FIGURE 1.2 – Exemple de l'industrie 4.0

1.2 Pourquoi l'interopérabilité est-elle importante ?

Les êtres humains sont de différentes origines, apparences, cultures, langues, etc. Cela les rend hétérogènes. Par exemple en figure 1.3, les personnes, qui parlent la même langue maternelle ou un langage commun, peuvent communiquer et se comprendre mutuellement. Cependant, il y a des personnes qui ne peuvent pas communiquer et se comprendre avec les autres parce qu'il est difficile de se conformer à une seule langue (native ou commune). Par conséquent, des moyens de traducteurs deviennent un besoin essentiel pour promouvoir l'interopérabilité entre eux. L'interopérabilité est simplement la capacité d'échanger et d'utiliser les informations (pour la définition plus détaillée voir la section 2.1). De même, les systèmes d'informations sont passés des systèmes monolithiques à des systèmes hétérogènes, autonomes, et distribués. L'hétérogénéité se produit en raison de la différence des langages de programmations, des matériels hardware, des plate-formes d'exploitations, des formats de données différents, etc. Même si chaque système peut fonctionner de façon autonome un moyen de collaboration et d'échange de données doit être un besoin essentiel entre ces systèmes afin de promouvoir l'interopérabilité entre eux.

Pour mieux comprendre le problème, prenons l'exemple en figure 1.4. Les systèmes hétérogènes autonomes sont distribués au niveau OT et IT (cf. section 1.1). Au niveau de OT, il existe de nombreux systèmes qui varient d'objets simples (capteurs et actionneurs) à des systèmes CPS complexes comme des Systèmes d'acquisition et de contrôle de données (Supervisory Control and Data Acquisition - SCADA en anglais). Un SCADA est un système de contrôle et d'acquisition de données contenant l'ensemble des éléments matériels et logi-

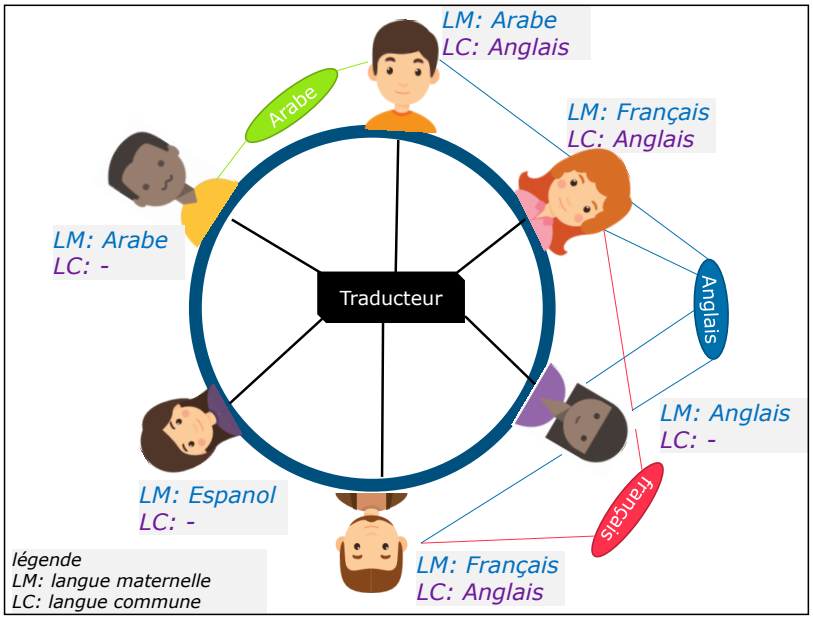


FIGURE 1.3 – Exemple de personnes hétérogènes

ciels permettant de traiter en temps réel et à grande échelle un grand nombre de télémesures, évènements de télégestion et de contrôler à distance des installations techniques et industrielles. Le SCADA se situe en haut de l'architecture du Contrôle-Commande d'un processus et représente l'échange vertical de données entre des installations techniques (I/O via PLC - Input/Output via Programmable Logic Controller en anglais) et l'interface homme-machine (Human Machine Interface - HMI en anglais). Les données sont échangées via des solutions Machine-To-Machine (M2M). Dans un environnement donné, différents systèmes SCADA peuvent coexister auprès de différents fournisseurs, chacun avec sa propre norme pour accéder, manipuler et représenter des données He and Xu [2014] tel que OPC UA Mahnke et al. [2009], MQTT [2], réseau basé sur la transmission radio comme Sigfox [3], etc. Chaque système fermé orienté est capable de travailler séparément et indépendamment, cependant il doit y avoir une collaboration entre eux.

Au niveau de IT, il y a d'autres systèmes d'information référés par les applications d'entreprise (Enterprise Application - EA - en anglais) [Izza, 2009]. Ce sont des logiciels pour gérer des tâches de traitement de l'information dans un domaine particulier au sein de l'entreprise telle que : le système gestion des processus industriels, Manufacturing execution systems -MES en anglais, qui est un logiciel pour tracer les données de production; le système **Gestion de maintenance assistée par ordinateur** (GMAO), Computerized Maintenance Management System - CMMS en anglais, qui est un logiciel destiné aux services de maintenance d'une entreprise. Il propose plusieurs fonctionnalités telles que la gestion des équi-

[2] http ://mqtt.org/
[3] http ://www.sigfox.com/

pements, la gestion de la maintenance, la gestion de la mise en sécurité des installations pour les travaux de maintenance, etc.; le système **Progiciels de Gestion Intégrés** (PGI), Enterprise resource planning - ERP en anglais, est un logiciel pour coordonner l'ensemble des activités d'une entreprise comme la production, l'approvisionnement, le marketing, la gestion des ressources humaines, etc.; la **gestion de la relation client** (GRC), Customer Relationship Management - CRM en anglais, est un logiciel pour gérer et rassurer les relations avec des clients; **Système d'Information géographique** (SIG), Geographic Information System - GIS, est un logiciel pour traiter, archiver, analyser et présenter les données spatiales et géographiques; **Systèmes d'aide à la décision** (SAD); le **Système prévision météorologie**; les **logiciels de simulation**; et beaucoup d'autres systèmes qui améliorent l'efficacité de l'entreprise. De nouveau, Chaque EA est capable de travailler séparément et indépendamment et elle est fournie par différents fournisseurs, en utilisant différentes normes, langage de programmation et protocoles [He and Xu, 2014].

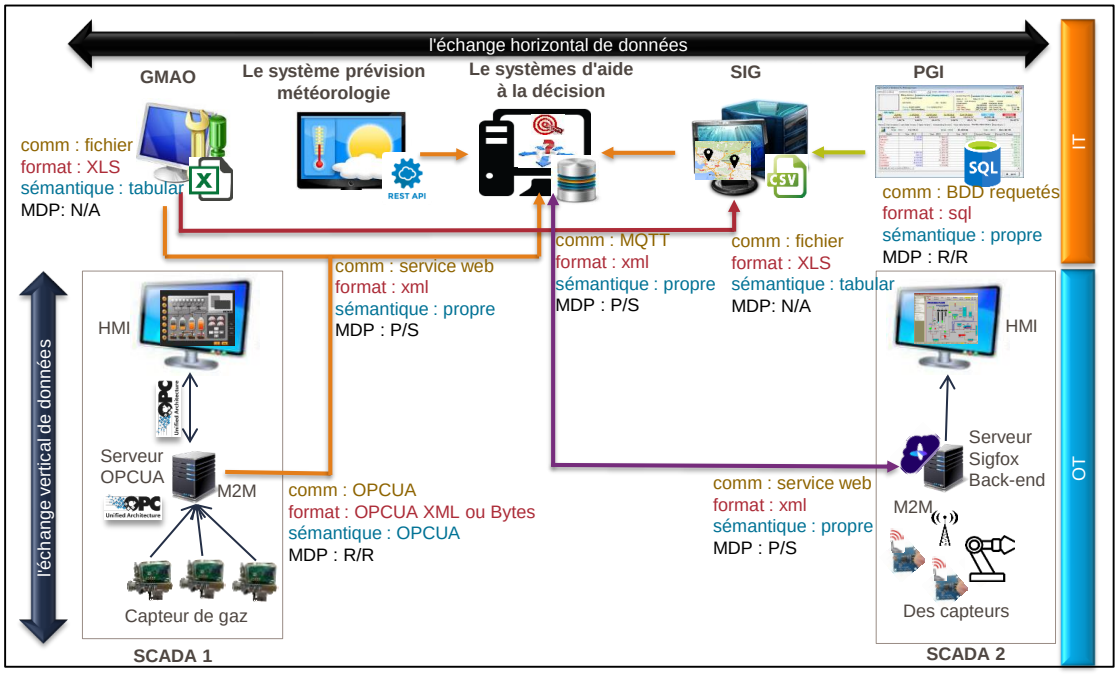


FIGURE 1.4 – Exemple d'un environnement hétérogène

Avec la troisième révolution industrielle, les systèmes sont connectés et communiquent au sein d'une structure hiérarchique. Grâce à la quatrième révolution, les OT et IT ne sont plus connectés en hiérarchie mais tous peuvent être connectés entre eux comme illustré dans la figure 1.5 [4]. Par exemple, un système peut fournir des données (Data Producer - DP) qui sont utilisées et consommées par d'autres systèmes d'information (Data Consumer - DC). Par exemple, le système SCADA génère une alarme (DP), cette alarme doit être affichée à l'emplacement approprié sur le système SIG (DC). Autre exemple, l'agrégation de

[4] https ://iot-analytics.com/industrial-internet-disrupt-smart-factory/

données à partir de plusieurs SCADA (DP) hétérogènes pour créer un cockpit de conduite hyperviseur (DC). En outre, le système d'aide à la décision (DC) doit prendre des décisions concernant la conduite de l'environnement. Cela nécessite l'agrégation de diverses données sources : topologie de réseau (SIG et/ou CMMS), les conditions météorologiques (prévision météorologique) et l'état des capteurs (SCADA). Pour conclure, les systèmes dans chaque niveau (IT/OT) peuvent échanger les données dans le même niveau et entre des niveaux. Par conséquent, les systèmes doivent être interopérables pour garantir l'échange d'informations.

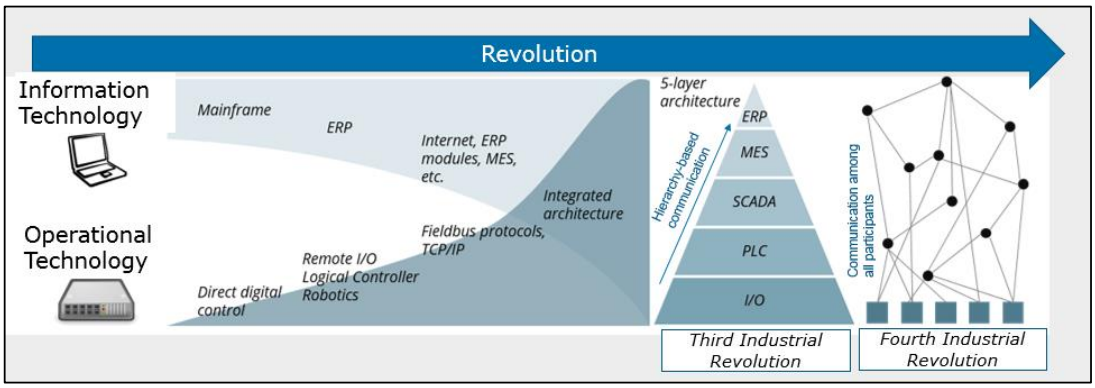


FIGURE 1.5 – Illustration de la convergence de l'IT et OT

L'interopérabilité joue un rôle important non seulement dans les systèmes d'information, mais aussi dans le hardware. Un exemple de solution pour l'interopérabilité hardware est la prise électrique pour un voyageur tel que montré à la figure 1.6. S'il voyage dans un pays qui utilise la même norme, son appareil électrique sera compatible. Mais s'il est dans un pays qui utilise une autre norme son appareil électrique ne devient pas compatible. Pour devenir interopérable, les solutions sont : (a) un adaptateur basique qui adapte la prise électrique d'un type vers un autre type ; (b) un adaptateur universel qui adapte la prise électrique de plusieurs types prédéfinis vers d'autres plusieurs types prédéfinis (c) un adaptateur universel plus générique, il a un noyau prédéfini dans lequel différents types de prises électricité peuvent être ajoutées.

FIGURE 1.6 – Exemple de solutions pour la prise électrique pour un voyageur

Les solutions d'interopérabilité apportées pour les hardwares sont aussi utilisées avec les systèmes d'information. Au même titre que le monde entier n'utilise pas la même norme unique pour la prise électrique, nous ne pouvons pas garantir un environnement qui suit la même norme unique dans le domaine de systèmes d'information. En fait, les normes ne sont pas suffisantes, car il n'existe pas un unique environnement qui suit la même norme [Lewis et al., 2008, Razzaque et al., 2016b, Terziyan et al., 2010]. Par analogie avec la métaphore de la prise électrique, il y a des solutions pour rendre l'environnement interopérable. Solution (a) correspond au développement ad-hoc spécifique, solution (b) correspond au développement entre les normes prédéfinies (tightly coupled), solution (c) représente une solution extensible et modulaire (loosly coupled).

Retournez à la prise électrique, supposons que votre appareil soit basé sur 220 V, mais vous avez voyagé dans un pays avec 110V, donc un autre adaptateur est nécessaire pour gérer le voltage ou l'adaptateur qui gère les différents points d'extrémité de la prise devrait avoir une unité de traitement de tension interne. La même chose se produit pour les systèmes d'information : les informations, qui sont fournies par un système, peuvent nécessiter un traitement avant d'être directement utilisées. Par exemple, le système GMAO définit les coordonnées en utilisant x, y, mais le système SIG utilise longitude et latitude. Un autre exemple, l'hyperviseur doit calculer la moyenne de pression de gaz et l'afficher, etc.

Pour résumer, avec la dernière révolution industrielle, tous les systèmes peuvent être connectés à tous les autres systèmes, donc tous devraient être capables d'échanger les données de façon interopérable. Une application sur le portable peut recevoir les données de plusieurs SCADA différents, le système d'aide à la décision peut commander les actionneurs, selon l'état de la situation, etc.

1.3 Quelles sont les causes des problèmes d'interopérabilité ?

L'agrégation et l'échange de données entre des systèmes hétérogènes peuvent être réalisés si et seulement si les systèmes sont interopérables. Cependant, les systèmes (logiciels et application) sont écrits dans différents langages de programmation fonctionnant sur différentes plates-formes avec différents systèmes d'exploitation et ont été développés par différents fournisseurs/individuels [Notkin et al., 1987]. Les systèmes utilisent leurs propres techniques, format et sémantique pour répondre à leurs besoins. Donc, les principales causes de problèmes d'interopérabilité peuvent être catégorisées du point de vue analytique de génie logiciel :

- **Différents mécanismes de communication** sont utilisés par les différents systèmes pour fournir, consommer ou échanger des données (ex. OPC UA, service Web Rest, partage de fichiers, Zigbee, etc.).

- **Divers façon d'interactions** sont utilisées pour échanger des données entre les systèmes. Certains systèmes producteurs de données (DP) (resp. consumateurs de données) fonctionnent sur l'option "Push", certaines sur l'option "Pull", d'autres dépendent d'intervalles de synchronisation périodique, etc.

- **Différents formats de données** sont utilisés par les systèmes (ex. le modèle de données OPC UA, les octets simples, les fichiers tabulaires CSV et Excel).

- **Différentes sémantiques de données** coexistent. En effet, chaque système a sa propre sémantique dans la production ou l'interprétation de données. Celles-ci peuvent aller de la dénomination simple (par exemple, les informations de géolocalisation utilisent le système de coordonnées différentes) à des différences structurales plus complexes.

- **Des règles métiers/opérationnelles** peuvent être nécessaires pour traiter les données d'entrée avant qu'elles ne puissent être présentées au consommateur (par exemple, la moyenne de valeur de plusieurs capteurs).

Les deux premiers problèmes nécessitent d'aborder l'interopérabilité au niveau technique [Franklin and Zdonik, 1998, Kubicek et al., 2011]. Le troisième problème aborde l'interopérabilité au niveau syntaxique alors que les quatrième et cinquième problèmes traitent l'interopérabilité au niveau sémantique [Kubicek et al., 2011, Sheth, 1999].

1.4 Notre question de recherche

Dans cette introduction, nous avons souligné que ces travaux sont dédiés à l'interopérabilité afin d'agréger et d'échanger des données entre des systèmes hétérogènes dont la conception oblige à surmonter quatre grands challenges : la diversité des mécanismes de communication, la diversité des formats de données, la diversité des sémantiques de données et le routage des données en temps réel. Cela fait naître un besoin théorique, méthodologique et technologique pour promouvoir l'interopérabilité entre des systèmes, ce qui nous emmène à la question de recherche : **comment aider les développeurs à assurer l'interopérabilité entre des systèmes hétérogènes en temps réel de manière efficace et moins chère, sans imposer une norme/technologie/domaine?** Afin de répondre à ces questions, nous effectuons d'abord un état de l'art sur des approches existantes. Nous postulons dans la suite du mémoire que les hypothèses suivantes sont vérifiées :

- H1 : l'interopérabilité entre les système hétérogènes peut être atteint en utilisant les concepts de la séparation de préoccupations et de l'ingénierie dirigée par les modèles.

- H2 : le coût et le temps pour promouvoir l'interopérabilité entre des systèmes peut être réduit en favorisant les caractéristiques de la réutilisabilité et de l'extensibilité.

1.5 Notre but et nos objectifs

Le but de cette recherche est d'assurer l'interopérabilité entre des systèmes hétérogènes en temps réel de façon **générique** (indépendante de domaine), **modulaire**, **agnostique** (n'impose aucune technologie ou norme) et **extensible** (pour s'adapter à n'importe quel mécanisme de communication, format de données et sémantique). Pour atteindre le but déclaré, cette thèse met en avant les objectifs suivants (cf. figure 1.7) :

- F1 - gérer plusieurs mécanismes de communication de façon agnostique et extensible (cf. sec. 3.3.1.1).
- F2 - gérer et intégrer diverses syntaxes de façon agnostique et extensible (cf. sec. 3.3.1.2).
- F3 - interpréter et intégrer diverses sémantiques en appliquant les règles métiers/opérationnelles sur les données de façon agnostique et extensible (cf. sec. 3.3.1.3 et 3.3.1.4).
- F4- acheminer (agréger et rendre disponible) les données en temps réel selon les options d'interaction (cf. sec. - 3.3.1.5).
- F5 - concevoir et développer un architecture et une système générique et modulaire pour assurer l'interopérabilité entre systèmes hétérogènes (cf. sec. 3.3.1).
- F6 - Développer une solution technique (un cadre) pour réaliser l'architecture proposée (cf. sec. 4.2).
- F7 - Développer un prototype et le tester sur une étude de cas réel (cf. sec.5.2).

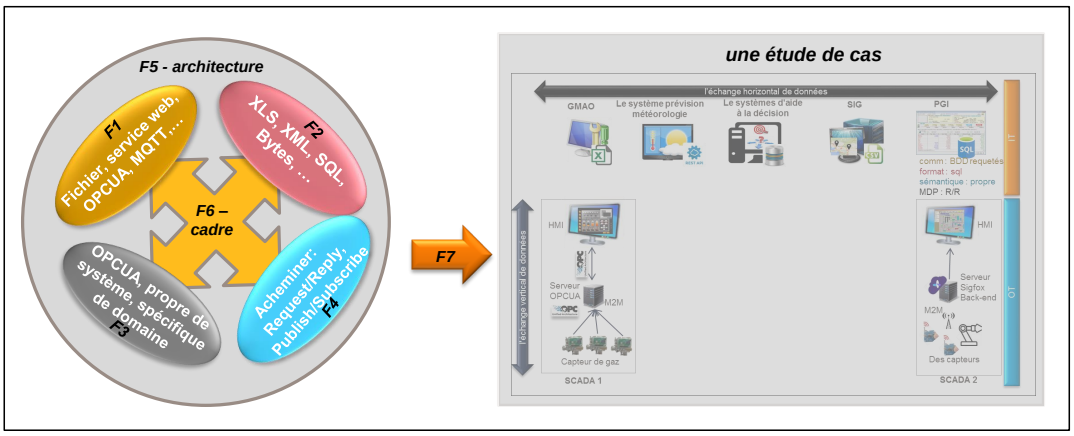


FIGURE 1.7 – Nos objectifs

1.6 Nos contributions

La contribution principale de ces travaux de recherche est la définition d'une architecture générique, théorique, pratique, modulaire, et extensible pour assurer l'interopérabilité entre des systèmes hétérogènes. Contrairement aux approches actuelles, cette recherche fournit une méthodologie qui repose sur les techniques de séparation profonde des préoccupations (separation of concerns en anglais) et des fonctions du système d'information. Elle est indépendante d'un domaine particulier et n'impose aucun standard ou norme pour les systèmes hétérogènes. Ensuite, cette méthodologie est réalisée en utilisant les concepts de l'ingénierie basée sur les modèles pour agréger les données venant des systèmes externes (producteurs de données - DP), les traiter et les rendre disponibles pour un autre système externe (consommateur de données - DC) de façon à faciliter l'intégration du système, à promouvoir l'échange interopérable des données. Dans notre approche, les systèmes (DP et DC) peuvent être intégrés selon leur propre mécanisme de communication, format et sémantique. Grâce à la séparation profonde des fonctions, les systèmes peuvent partager et utiliser les composants quand des niveaux systèmes sont intégrés. Les principales contributions de notre travail comprennent :

- Introduire une architecture et un cadre pour assurer l'interopérabilité entre des systèmes hétérogènes en temps réel. Cette nouvelle architecture est conçue pour soutenir l'interopérabilité de systèmes de façon : générique : à appliquer dans n'importe quel domaine ; modulaire : reposant sur les techniques de séparation profonde des fonctions du système d'information pour promouvoir la réutilisabilité ; agnostique (comme ils existent) sans restreindre les systèmes externes à une norme/technique particulière ; extensible : pour étendre le système pour l'adapter à toute technologie ;

- Mettre en œuvre l'architecture en utilisant les concepts de l'ingénierie basée sur les modèles, les approches orientées objet et des extensions pour promouvoir l'extensibilité. De cette façon, l'architecture peut combiner plusieurs mécanismes de communication, intégrer diverses syntaxes et sémantiques.

- Définir un modèle de données agnostique pour gérer plusieurs sémantiques.

- Définir un moteur pour orchestrer les données entre des systèmes en temps réel.

- Valider la solution dans un contexte réel (sec. 1.7)

Il existe cependant certaines limites à la recherche qui seront discutées dans le chapitre6

.

1.7 Contexte de travail

Ce travail est testé et illustré dans le cadre d'un projet national français financé par le FUI[5]. Ce projet (GONTRAND [6]) a pour but de répondre à un nouveau besoin technique provoqué par l'intégration croissante de biométhane dans les consommations d'une part, et à une efficacité croissante du réseau de distribution de gaz d'autre part. Il s'agit ici de deux des quatre macro-fonctionnalités définies par le "Groupe d'Experts 4" missionné en 2011 par la communauté européenne pour définir les bases du projet "Smart Gas Grids". Le projet GONTRAND vise à préparer, à moyen terme, les acteurs français à se positionner en leader sur le marché européen.

Il vise à améliorer l'efficacité de la distribution du gaz et d'accroître l'intégration de biométhane dans les réseaux de distribution. A ce titre, le système GONTRAND contribue à gérer en temps réel la quantité et la qualité du gaz à chaque point d'injection du réseau de gaz. Une architecture cible, fondée sur des briques innovantes en termes de choix logiciels et matériels, doit permettre un déploiement standardisé du système GONTRAND dans les systèmes d'information existants des gestionnaires de réseaux de distribution GDS, Régaz et GrDF.

Cinq briques sont définies : un analyseur gaz communicant, une plate-forme télécoms M2M, une plate-forme d'interopérabilité, un cœur de calcul et un cockpit de conduite (voir figure 1.8).

1. Un cockpit de conduite dédié à la visualisation, l'analyse et le pilotage du réseau en temps réel. Le cockpit de conduite vient en complément des SCADA existants. À ce titre, il assure un rôle d'hyperviseur.

2. Une plate-forme d'interopérabilité permettant la communication et l'échange de données sous des formats variés, issues de systèmes hétérogènes. Il permet d'assurer l'interopérabilité entre les systèmes d'information de GONTRAND et les systèmes d'information existants (SCADA, GMAO, SIG, etc.)

3. Un cœur de calcul destiné à l'aide de la décision basée sur : la modélisation et l'apprentissage des processus décisionnels et la modélisation dynamique du réseau et des points d'injection de biométhane. Le cœur de calcul dispose d'une interface avec le cockpit de conduite, permettant au cockpit de lancer des calculs, de récupérer des résultats et d'aider à la décision.

4. Une plate-forme M2M (Machine-to-Machine) capable de gérer des communications standards (2G et 3G) via des passerelles terrain (Gateway) et des communications radio bas débit.

[5] Fonds unique interministériel
[6] www.gontrand.net

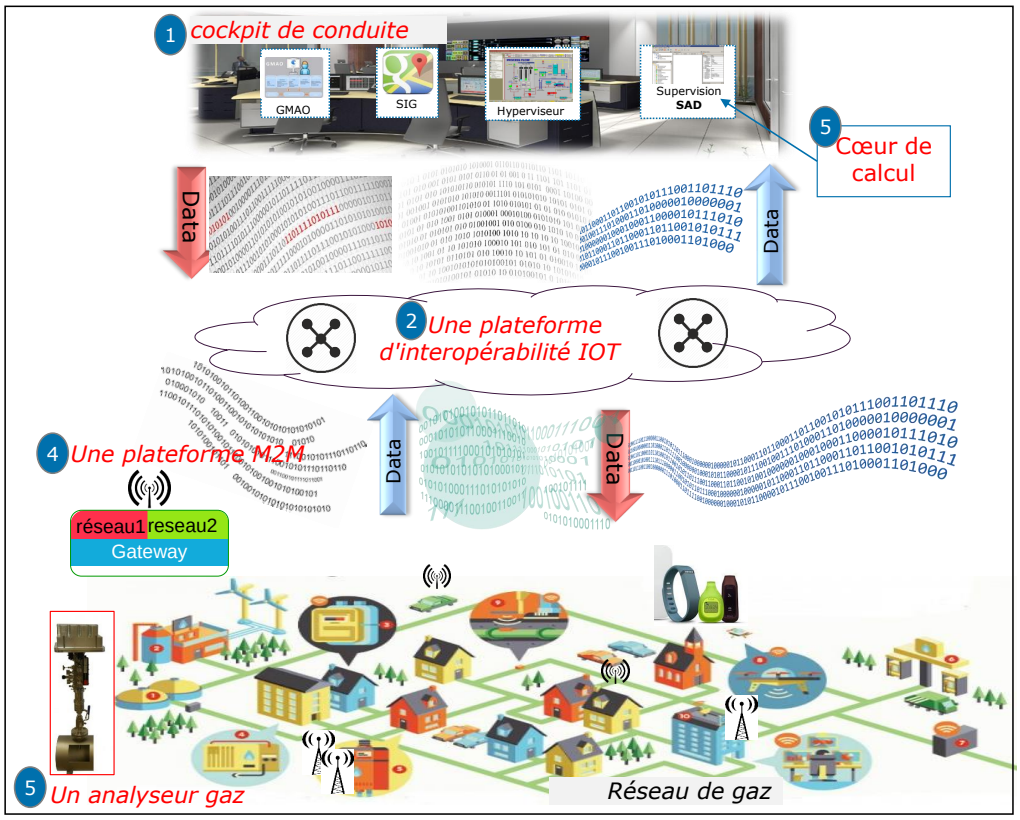


FIGURE 1.8 – Les briques de GONTRAND à développer

5. Un analyseur gaz permettant de mesurer in situ à distance la qualité du gaz et de transmettre les résultats à la plate-forme M2M via la Gateway.

Notre problématique de recherche scientifique émane du besoin d'étudier et proposer des solutions d'interopérabilités pour les domaines industriels. Il est validé avec le cas d'étude du projet GONTRAND (brique plate-forme d'interopérabilité) en particulier. Les activités de recherche en cours sont accomplies au sein du Laboratoire d'Ingénierie des Systèmes Physiques et Numériques (LISPEN EA 7515), localisé sur le campus Arts et Métiers ParisTech d'Aix-en-Provence et en partenariat avec l'équipe de développement et Innovation Sogeti HT - Capgemini, localisée sur le site d'Aix-en-Provence.

1.8 Conclusion

Dans ce chapitre, nous avons présenté les éléments qui ont donné lieu à notre travail. Aujourd'hui, la révolution industrielle a une grande influence sur la façon dont les systèmes collaborent afin de prendre des décisions importantes qui ont un impact sur l'entreprise. Tous les systèmes hétérogènes (au niveau IT et OT) doivent échanger les données avec d'autres systèmes hétérogènes (au niveau IT et OT) de façon interopérable en temps réel. Nous avons

présenté les principales causes de l'interopérabilité et nous avons décomposé le problème à plusieurs niveaux d'interopérabilité : technique, syntaxique et sémantique. Ensuite, nous avons présenté notre question de recherche, notre but, nos objectifs et nos contributions autour de l'interopérabilité entre les systèmes hétérogènes. Nous avons fermé le chapitre avec un aperçu du cas d'étude du projet GONTRAND dans laquelle notre travail sera testé et appliqué. Le chapitre suivant présente un état de l'art établi pour répondre à nos objectifs et contributions dans ce travail de recherche.

Chapitre 2

Etat de l'art

Sommaire

Les systèmes (au niveau OT et IT) ne peuvent pas coexister de manière isolée, mais exigent que leurs données soient partagées de manière à accroître la productivité de l'entreprise. Le but de ce chapitre est de présenter un état de l'art afin de répondre à nos objectifs et contributions autour de l'interopérabilité. Les solutions d'interopérabilité peuvent s'appuyer sur des études spécifiques au niveau de l'interopérabilité (technique, syntaxique, sémantique), des solutions architecturales, des normes ou des middlewares.

Ce chapitre est divisé en trois sections principales : nous présentons d'abord le travail existant axé sur chacun des trois niveaux d'interopérabilité, puis nous présentons des solutions et des approches plus générales qui peuvent être considérées dans notre contexte tels que les normes, les cadres d'architecture, les middlewares, les solutions académiques et commerciales. Ensuite, nous présentons les concepts de l'ingénierie dirigée par les modèles et comment nous pouvons les utiliser pour atteindre notre objectif.

2.1 L'interopérabilité et ses niveaux

Il y a plusieurs définitions pour le terme "interopérabilité". Deux définitions importantes de l'interopérabilité sont citées dans ce travail. Le premier est tiré du glossaire mondial standard de l'IEEE (Institute of Electrical and Electronics). Il définit l'interopérabilité comme la capacité de deux ou plusieurs systèmes ou composants d'échanger et d'utiliser les informations qui ont été échangées sans aucune intervention [IEE, 1991]. Une autre définition a été donnée par [Wegner, 1996] avec un sens plus profond. Il définit l'interopérabilité comme l'aptitude de plusieurs systèmes à communiquer, coopérer et échanger des données et services, malgré les différences dans les implémentations ou les modèles d'abstraction.

L'interopérabilité existe à plusieurs niveaux [Hans van der Veer, 2008, Kubicek et al., 2011, Sheth, 1999] :

- L'interopérabilité technique
- L'interopérabilité syntaxique
- L'interopérabilité sémantique

Ces trois niveaux correspondent respectivement à nos quatre premiers objectifs (F1, F2, F3, F4 - cf. section 1.5). Dans la partie suivante, nous présenterons l'état de l'art pour aborder chaque niveau.

2.1.1 L'interopérabilité technique

L'interopérabilité technique représente la communication et l'interaction entre des machines, périphériques et des applications en utilisant des composants matériels/logiciels différents, des systèmes différents, des plates-formes, des protocoles différents et des normes différentes [Hans van der Veer, 2008]. L'interopérabilité technique peux être classifiée à deux

niveaux : les mécanismes de communication et les paradigmes d'interaction qui correspondent à notre objectifs F1 et F4.

2.1.1.1 Les mécanisme de communication

Les mécanismes de communication garantissent la transmission correcte des bits. Les différents mécanismes sont définis par des protocoles. Ici, nous entendons les protocoles de dernière couche (Application) du modèle OSI (Open System Interconnection en anglais) [Alani, 2014, Zimmermann, 1980]. Ce modèle, qui sert de base aux réseaux de télécommunication et qui gère l'interopérabilité de la communication, définit comment deux systèmes communiquent via sept couches (physical, data link, network, transport, session, presentation et application) tel qu'illustré dans la figure 2.1. Les cinq premières couches se concentrent sur la façon de transmettre les données entre deux appareils via différents types de réseaux de télécommunication. Bien que les deux dernières couches soient concernées par la syntaxe et la sémantique des données, elles assurent seulement que les données sont transmises avec les syntaxes et les formats appropriés [Venot et al., 2014]. Mais il ne définit pas comment les différentes applications vont interpréter ces données. Dans cette dissémination, nous traitons le problème d'interopérabilité depuis la dernière couche de la pile de communication, c'est-à-dire au niveau de la couche application, qui est visible à l'utilisateur final en utilisant différents protocoles tels que HTTP, OPC, FTP, MQTT, Zigbee etc. Les détails de ce modèle sont hors de portée de ce travail.

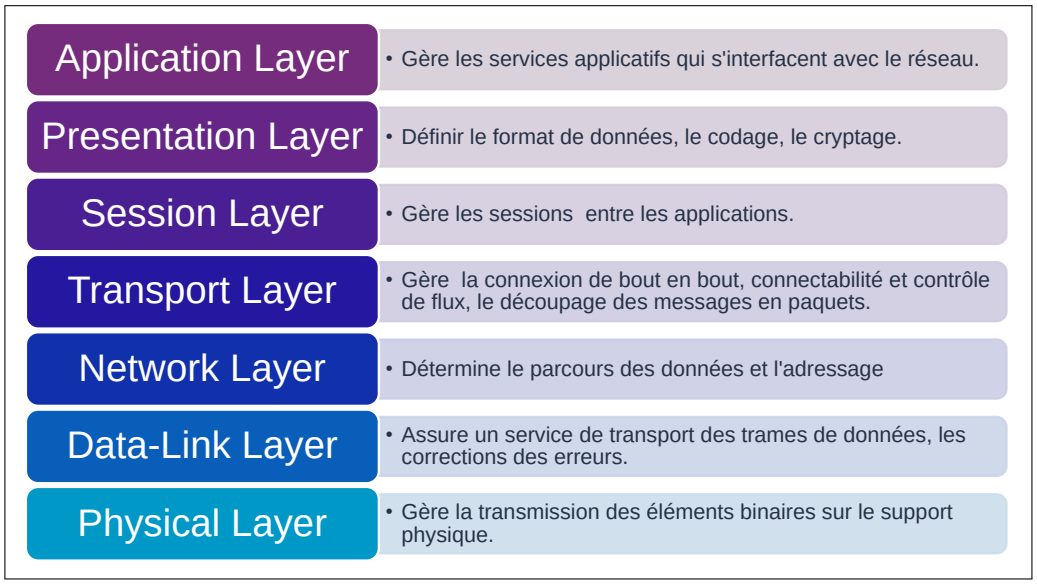


FIGURE 2.1 – Modèle OSI

OPC (OLE for Process Control) est l'une des protocoles/normes largement utilisé pour communiquer dans la couche d'automatisation dans tous les types d'industries. En plus de la communication, l'ancienne version intitulée "OPC Data Access" dépendait de la plate-

forme et utilisait la technologie Microsoft COM/DCOM. Cependant, la nouvelle version OPC Unified Architecture (OPC UA) est indépendante de la plate-forme et elle sert aussi à la modélisation de données [Mahnke et al., 2009]. Hyper Text Transport Protocol (HTTP) est un protocole de communication pour le World Wide Web. Au début, il a été développé pour l'envoi d'hypertexte sur Internet, puis il a été utilisé pour plusieurs objectifs comme l'application Web et la communication M2M. PROFIBUS [1] et Modbus [2] sont des normes de communication qui ont connu un grand succès dans un large domaine d'application pour une production efficace et économique. Message Queuing Telemetry Transport (MQTT) [Banks and Gupta, 2014] est un protocole de messagerie (M2M) léger réalisé par IBM. Il facilite la transmission des données de télémétrie pour les clients réseau dont les ressources sont limitées. Il joue un rôle important dans l'IdO et l'IdO industriel et il suit un modèle de communication de type publish/subscribe. Constrained Application Protocol (CoAP)[Bormann et al., 2012] est une autre norme de messagerie utilisée dans les réseaux de capteurs sans fil pour former l'Internet des objets et il suit un modèle de communication de type Request/Reply. Messages radio utilisés par des réseaux comme Sigfox [3]. Sigfox a adapté un protocole léger pour traiter de petits messages qui utilisent une bande ultra étroite. D'autres exemples de protocoles incluent Advanced Message Queuing Protocol (AMQP) [Vinoski, 2006], Data Distribution Service (DDS) [Pardo-Castellote et al., 2005], etc.

À mesure que la multiplicité et l'hétérogénéité des systèmes augmentent, la complexité de l'intégration de tels systèmes est augmenté considérablement. En effet, les environnements hétérogènes nécessitent souvent une combinaison de ces mécanismes pour que les systèmes communiquent entre eux. Un certain nombre de cadres ont été réalisés pour la connectivité entre périphériques et applications. IOTivity [4] adopte le protocole d'application contraint (CoAP) comme protocole de connectivité IdO principal et permet l'extension vers d'autres protocoles de communication (MQTT, HTTP, etc.) en utilisant des plug-ins de protocole. De même, AllJoyn [5] est un framework logiciel open source pour la communication et la découverte entre les périphériques. Le cadre de Thread [6] groups définit un protocole basé sur IPv6 pour la communication entre les périphériques, en particulier pour la domotique.

2.1.1.2 Les paradigmes d'interaction

Les différentes technologies et mécanismes de communication reposent sur quatre modèles d'interaction tels qu'illustrés dans la figure 2.2 [Mühl et al., 2006]. Ces modèles déterminent les différentes manières d'établir des méthodes d'interaction entre différents homologues (systèmes, processus ou composants). En effet, toute interaction entre différents pairs peut

[1] https ://www.profibus.com
[2] http ://www.modbus.org/
[3] https ://www.sigfox.com/en
[4] https ://www.iotivity.org/
[5] https ://allseenalliance.org/framework
[6] http ://threadgroup.org/

être catégorisée selon les différents modèles qui sont eux-mêmes classés selon deux attributs : initiateur et adressage. Le premier attribut, initiateur, détermine qui initie l'interaction : (i) un fournisseur de données ou de fonctionnalité ; (ii) un consommateur de données ou de fonctionnalité. Le deuxième attribut, adressage, détermine si l'adresse des autres pairs est connue ou inconnue.

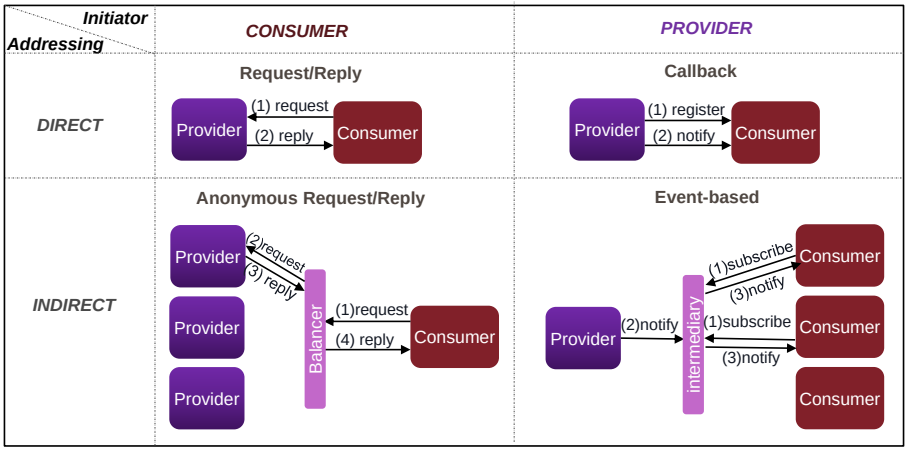


FIGURE 2.2 – Les modèles d'interaction

Request/Reply

C'est le modèle d'interaction largement connu, où un consommateur (initiateur) demande des données ou une fonction à un fournisseur. Des exemples d'un tel modèle d'interaction sont des interactions client/serveur, des appels de procédures distantes, et des services web s'appuyant sur des requêtes telles que SOAP, etc. Dans le modèle d'interaction client/serveur, le consommateur est un client qui initie la demande et le fournisseur est le serveur qui fournit la réponse. Dans ce modèle, on connaît l'identité du fournisseur qui doit répondre avec des données ou des fonctionnalités requises.

Anonymous Request/Reply

C'est un cas particulier du modèle d'interaction Request/Reply où l'identité du fournisseur n'est pas connue à priori. En fait, la requête est envoyée à un ensemble arbitraire de fournisseurs et le consommateur suppose recevoir au moins une réponse. Exemple d'un tel modèle peut-être vu dans l'environnement d'équilibrage de charge (load balancing) avec des fournisseurs redondants.

Callback model

Un consommateur s'enregistre lui-même auprès des fournisseurs pour être intéressé par une notification. Cette notification est déclenchée par les fournisseurs, si une condition observée

devient vraie. En d'autres termes, le fournisseur évalue à plusieurs reprises la condition et s'il est satisfait, il notifie le composant enregistré. Les fournisseurs ici sont l'initiateur de la fonctionnalité de données et de notification. L'identité des composants est connue à priori afin d'établir l'interaction. Le motif de conception "observer pattern" est un exemple largement utilisé de ce type d'interaction.

Event-Based

Le fournisseur de données initie la communication en produisant des notifications qui ne sont pas adressées à un consommateur spécifique. Les consommateurs émettent plutôt des abonnements qui décrivent le type de données qui les intéresse. Une notification est envoyée au consommateur s'il correspond à l'un de ses abonnements via un service de messagerie intermédiaire. Les fournisseurs ne connaissent pas les consommateurs, ils n'ont donc pas besoin de gérer les inscriptions d'abonnement. Le mécanisme de communication publish/subscribe est un exemple de ce type d'interaction. Techniquement, il n'y a pas de meilleure solution pour représenter ces paradigmes, mais cela dépend de l'environnement de déploiement et de la qualité, de la flexibilité requise [Fiege, 2005]. Les travaux de [Franklin and Zdonik, 1998] fournissent une classification plus détaillée afin de prendre en compte les différents modes d'interaction. Les deux modèles Resquest/Reply et Autonomous Request/Reply sont basés sur le modèle de diffusion de données "Pull". Ici, le consommateur (client) lance la demande de données du fournisseur (serveur). Cependant les modèles Call-Back et Event-based sont basés sur le modèle "Push". Ici, le fournisseur de données (serveurs) met à disposition ses données. Cependant, cette classification de "Push vs Pull" est une seule dimension de la diffusion de données. Elle est basée sur qui a lancé la transmission de données. Cependant, il y a plus de dimensions de la classification de diffusion de données. En effet, les actions de Push et Pull peuvent être effectuées de façon apériodique ou périodique qui résulte dans une autre dimension "Apériodic vs Periodic". L'approche Apériodic est axée sur les événements, le client (pull) déclenche la demande de données et le serveur (push) fourni la mise à jour de données. En revanche, l'approche Périodic est effectuée selon une fréquence prédéfinie. En ce qui concerne la derniere dimension, il classe la diffusion en communication Unicast ou 1-to-N. En communication Unicast, les données sont diffusées d'un fournisseur vers un consommateur. Cependant, avec la communication 1-to-N les données sont diffusées d'un seul fournisseur vers plusieurs consommateurs. Ces dimensions de diffusion peuvent être mélangées et entraîner les options suivantes telles qu'illustrées dans le figure 2.3 [Franklin and Zdonik, 1998].

- Aperiodic Pull : il s'agit du traditionnel mécanisme request/response ou un consommateur (client) demande (pull) les données sur une connexion unicast. Si la connexion est 1-to-N, les clients qui n'ont pas généré la demande peuvent espionner 'snoop' la ré-

ponse du client qui a déclenché la demande [Acharya et al., 1997, Aksoy and Franklin, 1999, Liebig et al., 2000].

- Periodic Pull : les demandes (pulls) sont lancées périodiquement. Si la connexion est unicast, le client qui a effectué le demande reçoit la réponse. Si la connexion est 1-to-N, les clients qui n'ont pas généré la demande peuvent espionner 'snoop' la réponse du client qui a déclenché la demande.

- Aperiodic Push : il représente la communication publish/subscribe [Eugster et al., 2003, Glance, 1996]. Des consommateurs sont abonnés à des données qui sont fournies par des fournisseurs. À chaque fois que de nouvelles données arrivent, elles sont poussées (push) par le fournisseur et mises à disposition pour le destinataire. Aperiodic Push peut être réalisé pour les deux types de connexion unicast et 1-to-N.

- Periodic Push : c'est un cas particulier de "Aperiodic Push" où les données sont poussées périodiquement et mises à la disposition des consommateurs.

Les pros et cons de chaque approche ne font pas partie de l'objectif de cette thèse. Plus de détail sont disponibles dans ces travaux [Fiege, 2005, Franklin and Zdonik, 1998, Mühl et al., 2006]. En fait, il y aura un mélange de ces approches dans un environnement hétérogène, donc nous devons interagir avec toutes ces approches.

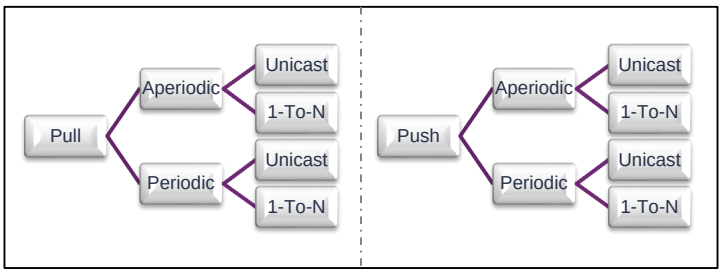


FIGURE 2.3 – Les options de diffusion de données [Franklin and Zdonik, 1998]

2.1.2 L'interopérabilité syntaxique

Après la communication entre des machines et des applications, les messages doivent avoir une bonne syntaxe et un bon encodage. Le syntaxique concerne les formats des données et en informatique la syntaxe se réfère à la grammaire et aux règles formelles pour définir des ensembles de données indépendamment de leur signification. Les différents systèmes utilisent des formats différents pour représenter leurs données, ce qui entraîne le problème de l'interopérabilité syntaxique. Plusieurs formats d'échange et normes existent tels que Extensible Markup Language (XML) [Bray et al., 1997], JavaScript Object Notation (JSON) [Crockford, 2006], Comma-Separated Values (CSV) [Shafranovich, 2005], base de données, etc. Chacun a ses propres caractéristiques en termes de lisibilité, d'extensibilité, de validation, de

support pour la structure des données, etc. Quelques exemples de travaux sur la comparaison en termes de performance entre XML et JSON sont donnés par [Nurseitov et al., 2009, Sumaray and Makki, 2012].

Dans le paragraphe suivant, nous présenterons une brève introduction non exhaustive de formats et normes.

2.1.2.1 XML

XML est un langage basé sur l'utilisation de balises définissant un format universel de représentation des données qui est standardisé par le World Wide Web Consortium. (W3C) [Bray et al., 1997]. L'un des objectifs d'échanger les informations sur le Web. Il est caractérisé par la simplicité, la lisibilité humaine, l'indépendance de la plate-forme, l'extensibilité et la transmissibilité entre des environnements sans perdre des informations. L'extensibilité permet aux utilisateurs de déclarer et d'utiliser leurs propres balises et attributs. La structure de données et sa syntaxe sont décrites par les spécifications XML Schema Definition (XSD) qui a été introduit en remplacement de Document Type Definition (DTD). Donc, les documents XML peuvent valider sa structure de contenu en comparant avec sa XSD/DTD. Un exemple de document XML qui permet de décrire une liste des employeurs est illustré dans figure 2.4.

```
<employees>
    <employee>
        <firstName>Ahmed</firstName>
        <lastName>Yousef</lastName>
    </employee>
    <employee>
        <firstName>Peter</firstName>
        <lastName>Smith</lastName>
    </employee>
</employees>
```

FIGURE 2.4 – Exemple d'un document XML

2.1.2.2 JSON

JSON [Bray, 2014, Crockford, 2006] est un format léger pour la représentation et l'échange de données. Il est facilement utilisé, analysé et écrit par des ordinateurs et des humains. Il est basé sur un sous-ensemble du langage de programmation JavaScript [Assembly, 1999]. La structure JSON est composée d'une liste ordonnée des objets (un tableau) ou chaque objet est une collection de couples nom/valeur séparés par des deux-points " :". La figure 2.5 illustre un exemple de fichier JSON pour décrire la liste des employeurs.

FIGURE 2.5 – Exemple d'un document JSON

2.1.2.3 XMI

XMI (XML Metadata Interchange) [Rys et al., 2009], qui est une norme OMG, fournit un moyen pour échanger les informations de métadonnées sur XML. Il permet l'échange des structures de données, leurs propriétés, les relations, etc. entre plusieurs outils de modélisation en utilisant les langages comme UML (Unified Modeling Language) et Metaobject Facility (MOF), etc. XMI permet d'ajouter des informations supplémentaires (par exemple, des contraintes sur les données ou des contraintes sur la structure des données) ce qui n'est pas possible avec XML. Plus de détails sur les aspects de modélisation sont expliqués dans la section 2.3.

2.1.2.4 Les fichiers STEP

ISO 10303, aussi appelée STEP (STandard for the Exchange of Product model data), est une norme pour l'échange de données de produits [Pratt, 2001a]. Il est largement utilisé dans l'interopérabilité de logiciels de conception assistée par ordinateur (CAO) pour échanger les modèles de produit entre plusieurs logiciels. STEP est composé de plusieurs parties dans lesquelles il y a la partie 21 (STEP File) et la partie 28 (STEP-XML) qui sont concernés par la représentation et le format de données. La première utilise le codage en clair de la structure d'échange alors que la seconde utilise le codage XML. Cependant, il y a d'autres parties concernant les schémas comme EXPRESS qui sont hors de portée de cette section.

Les formats décrits ci-dessus sont les différentes façons de représenter et formater les données. Certains sont dépendants du domaine pour les données de produit, d'autres sont indépendants du domaine comme XML. XMI est une application de XML pour l'échange de métadonnées. Il existe des outils/techniques qui fournissent une conversion entre des

formats tels que JSON vers XML [7] et vice versa, CSV vers JSON, SQL vers JSON [8], XSLT (eXtensible Stylesheet Language Transformations) pour transformer des fichiers XML vers d'autre format [Clark et al., 1999], etc. Les environnements hétérogènes doivent interagir avec ces formats, mais le problème d'interopérabilité ne s'arrête pas ici et il monte au niveau sémantique.

2.1.3 L'interopérabilité sémantique

Même si les données sont disponibles et bien structurées après la communication, ces données ne sont utiles que si l'autre système est conscient de l'information de ces données, c'est-à-dire, l'autre système doit interpréter et comprendre les données afin de les utiliser. Ainsi, l'interopérabilité sémantique est définie comme la capacité de deux systèmes ou plus à échanger des informations et à faire en sorte que la signification de ces informations soit automatiquement interprétée par le système de réception avec suffisamment de précision pour produire des résultats utiles pour les utilisateurs finaux des deux systèmes [Ceusters and Smith, 2010]. Autrement dit, l'interopérabilité sémantique se réfère à la capacité de deux ou plusieurs systèmes à échanger des informations avec une signification non ambiguë et partagée [Iroju et al., 2013]. En fait, l'interopérabilité nécessite des accords sémantiques en complément de syntaxiques pour réussir[Bastida et al., 2008]. L'interprétation correcte est l'élément clé pour l'interopérabilité sémantique. Elle est abordée de manière différente. Quelques travaux utilisent les langages de modélisation pour définir les modèles de données et d'autres s'appuient sur des ontologies de domaine.

Les langages de modélisations incluent : (i) Unified Modeling Language (UML) [Rumbaugh et al., 2004], qui est un standard de l'OMG, est un langage de modélisation, objet-orienté, graphique et textuel pour modéliser les multiples aspects d'un système d'information. Ce modèle serve à réaliser et développer les systèmes d'information. Il est composé de plusieurs sous-langages pour modéliser les aspects structurels et comportementaux d'un système d'information. Il est largement utilisé dans l'industrie pour la modélisation des données [Muller, 1999] en utilisant spécialement le diagramme de classe. Il utilise les concepts de la programmation orientée objet comme classes, attributs et associations pour modéliser un système. (ii) (Meta-Object Facility) MOF est un langage de modélisation non seulement pour UML mais aussi pour d'autres langages comme Common Warehouse Metamodel (CWM). Les modèles sont sérialisés en utilisant XMI discuté précédemment. Plus des détails sur les aspects de modélisation seront présentés dans la section 2.3. (iii) Domain specific languages (DSL) sont dédiés pour modéliser un domaine spécifique. Quelques travaux utilisent des DSL pour gérer les sémantiques telles que Common Information Model (IEC 61970, 61968, 62325) [Uslar et al., 2012] qui définit les composants des systèmes d'alimentation électrique et leurs

[7] http ://json2xml.com/
[8] http ://www.csvjson.com/

relations, Pipeline Open Data Standard (PODS) [McCallum, 2000] pour l'industrie du pipeline, Gas Distribution Model (GDM) [9] pour représenter les composants du réseau de gaz, etc.

D'autres travaux s'appuient sur des ontologies de domaine [Bittner et al., 2005, Zang and Pohl, 2003]. Une ontologie est *une spécification formelle et explicite d'une conceptualisation partagée* [Gruber, 1993]. ISO-15926 fournit une ontologie pour l'industrie pétrolière et gazière. D'autres ontologies de domaines incluent BBO [10] pour le domaine biologique et biomédical, [Lin and Harding, 2007] pour l'ontologie d'ingénierie des systèmes de fabrication, etc. Ces ontologies de domaine et la standardisation du modèle de données ont apporté la convergence sémantique pour l'échange d'informations, mais le manque d'un environnement conforme aux accords communs et au modèle de données reste la principale charge pour l'interopérabilité.

OPC UA [Mahnke et al., 2009] en plus de ses mécanismes de communication, fournit un moyen générique de créer des modèles de données. OPC UApermet la création des modèles qui sont indépendants de domaine ainsi qu'open Data Format (O-DF) et Common Data Model (CDM) [Nativi et al., 2008, Robert et al., 2016]. O-DF fait partie de la norme "Open Group Internet of Things" pour représenter de façon générique les informations des objets (cf. figure 2.6 [11]). O-DF utilise la notion générique d'Objets où il contient un objet racine comme élément supérieur qui contient des sous-éléments Object en hiérarchie. Les objets sont identifiables et peuvent également avoir des propriétés, qui sont des sous-éléments appelés InfoItem, ainsi que des sous-éléments Objets. Les infoItem sont utilisés pour gérer les valeurs et les meta données. De même CDM est un modèle de données abstrait pour les ensembles de données scientifiques. Un extrait de ce modèle est illustré dans figure 2.7. Il est très générique et indépendant de domaine spécifique. Nous sommes intéressés par les concepts suivants : **Group**, un conteneur pour d'autres concepts tels que des variables et des sous-groupes imbriqués ; **Variable**, un conteneur de données qui est caractérisé par un nom, un type de données et éventuellement un ensemble d'attributs ; **Attribute**, une méta donnée pour caractériser une variable ou un groupe. Il est caractérisé par un nom, un type de données et une valeur.

Il existe différentes façons d'interopérer différents modèles de données. Quelques travaux sont basés sur le mapping (faire correspondre) des modèles de donnée tels que [Rohjans et al., 2013] pour faire correspondre les modèles de données dans le domaine de l'électricité CIM vers OPC UA. Selon les normes ISO 14258 :1998, il y a trois approches distinctes pour atteindre l'interopérabilité sémantique [Organizatio, 1998, Paviot et al., 2011] (cf. figure 2.8) : **l'intégration, l'unification et la fédération**. L'approche d'intégration est basée sur l'idée de se conformer à un seule modèle de données. Un consensus doit être trouvé entre tous les acteurs (systèmes) pour définir ce modèle intégrant tous les modèles. L'approche d'unification

[9] http ://gtigpsresearch.blogspot.com/p/gas-distribution-model.html
[10] http ://www.obofoundry.org/
[11] http ://www.opengroup.org/iot/odf/p1.htm

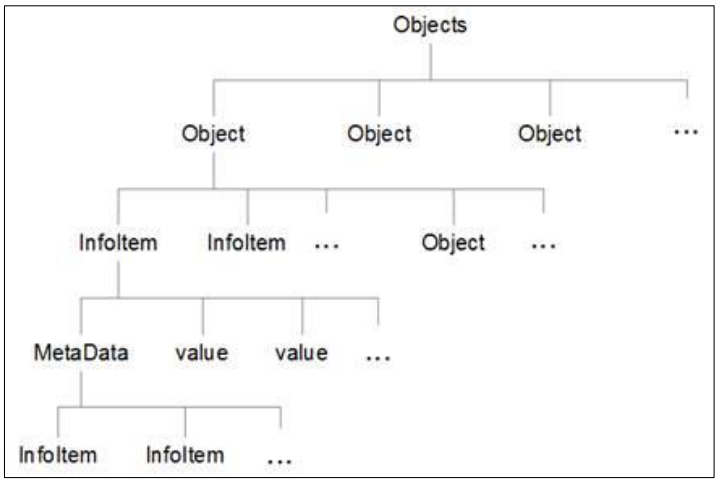


FIGURE 2.6 – Illustration de la hiérarchie des éléments O-DF

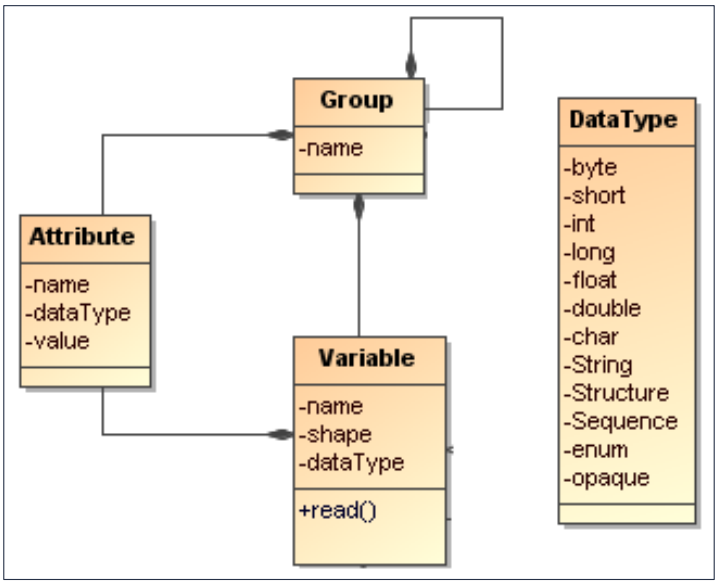


FIGURE 2.7 – Modèle de données CDM [Nativi et al., 2008]

vise à proposer un modèle médiateur (c.-à-d. pivot) utilisé pour lier les différents modèles différents. La liaison a besoin de spécifier les règles de mapping pour définir les associations sémantiques. L'approche de fédération consiste à associer plusieurs modèles de données distincts de façon dynamique. Les modèles de chaque acteur (système) sont liés directement via les règles de mapping et avec des méthodes heuristiques pour détecter les couples de concepts liés au niveau sémantique (similarité ou équivalence).

2.1.4 Synthèse

Nous synthétisons les solutions d'interopérabilité à différents niveaux pour gérer l'interopérabilité au niveau communication, syntaxique et sémantique tel qu'illustré dans la figure

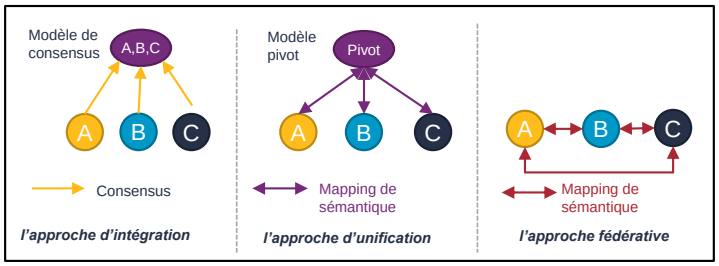


FIGURE 2.8 – Exemples des approches d'interopérabilités sémantiques

2.9. Au niveau technique, les solutions dépendent de l'imposition d'un mécanisme de communication principal comme [Banks and Gupta, 2014, Bormann et al., 2012, Mahnke et al., 2009]. Pourtant, dans le monde de systèmes hétérogènes, il faut interagir avec différents mécanismes de communication. Donc, une approche pour combiner plusieurs mécanismes devient essentielle. Ces mécanismes de communication servent à communiquer et diffuser les données entre les systèmes. Les différents mécanismes de communication reposent sur la liste définie des options d'interaction[Franklin and Zdonik, 1998].

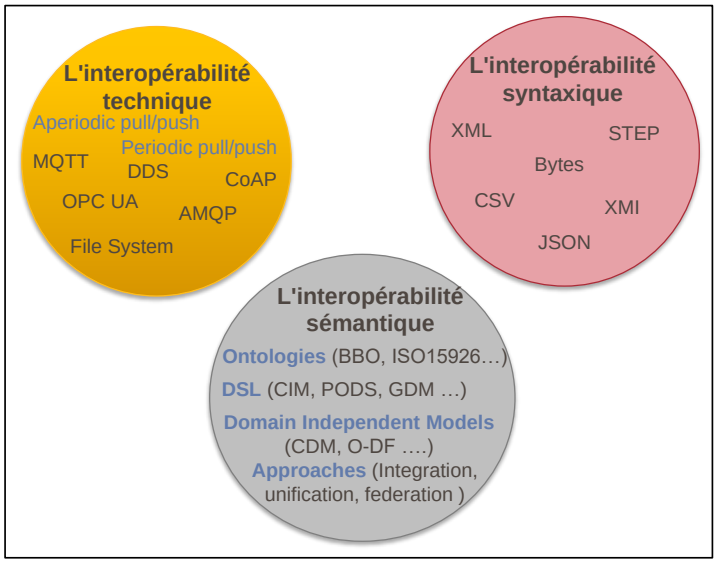


FIGURE 2.9 – L'interopérabilité et ses niveaux

Au niveau d'interopérabilité syntaxique, nous ne pouvons pas encore garantir que tous les systèmes soient conformes à une syntaxe unique comme [Bray et al., 1997, Crockford, 2006, Pratt, 2001a, Rys et al., 2009]. Quelques solutions proposent la conversion entre les syntaxes comme [Clark et al., 1999]. Cependant, elles sont limitées à quelques formats. Il faut proposer une approche qui permette de faire la conversion de n'importe quel format vers n'importe quel format. XMI s'est avéré très utile dans le monde IDM.

Au niveau de l'interopérabilité sémantique, les solutions proposent d'utiliser les modèles de données dépendants de domaines ou les ontologies. Si une caractéristique du modèle

de données ou l'ontologie n'est pas suffisante, le modèle doit être étendu ou modifié. Cela conduit à des nombreux modèles de données/d'ontologies étendus à chaque fois qu'une nouvelle exigence est introduite Ellis et al. [2017]. De plus, il est difficile de trouver un modèle de données basé sur l'approche d'intégration pour fusionner tous les modèles. Nous nous sommes basés sur l'idée d'avoir un modèle neutre, indépendant de domaine comme dans [Nativi et al., 2008, Robert et al., 2016]. L'approche d'unification [Organizatio, 1998, Paviot et al., 2011] sera essentielle pour faire le mapping entre les sémantiques de systèmes hétérogènes et un modèle pivot (indépendant de domaine). De plus, une approche d'intégrer plusieurs règles de mapping entre les modèles de domaine et le modèle pivot sera nécessaire.

Les solutions présentées ci-dessus gèrent chaque niveau séparément. Dans la section suivante, nous présentons les solutions qui prennent en compte plusieurs niveaux d'interopérabilité ensemble.

2.2 Approches de l'interopérabilité multi niveaux

Dans cette partie, nous faisons un tour sur les solutions pour répondre à notre cinquième objectif F5. Nous commençons par les solutions traditionnelles "ad-hoc", après nous discutons des solutions basées sur des normes, des architectures, des middlewares et des solutions académiques et commerciales.

2.2.1 Les solution ad-hoc

Quelques industries traitent l'interopérabilité en utilisant des solutions ad-hoc. Ces solutions ad-hoc sont basées sur l'utilisation de codage en dur (Hard coded solutions) pour réaliser une mapping à double sens entre deux systèmes. Cette approche est également référée par le pontage (Bridging en anglais) [Wegner, 1996]. Ce mapping est une sorte de médiateur logiciels pour définir des passerelles entre les applications permettant de promouvoir l'interopérabilité entre des systèmes hétérogènes [Izza, 2009]. Cependant, le nombre de mapping augmente avec le nombre de systèmes intégrés dans l'environnement qui à son tour devient coûteux et complexe à gérer comme illustré dans la figure 2.10.

2.2.2 Normes

Beaucoup d'organisations s'attaquent au problème d'interopérabilité en utilisant des normes qui sont réparties à travers les niveaux d'interopérabilité. Nous avons déjà présenté les normes dans chaque niveau, par exemple, les normes COAP, OPC UA , etc. au niveau technique; les normes XML, XMI au niveau syntaxique; les normes OPC UA, MOF, UML, etc. au niveau sémantique. Ici, nous présentons aussi d'autres normes dépendantes de domaine qui peuvent traiter un ou plusieurs niveaux l'interopérabilité. ISO-15926 fournit une fondation pour la modélisation des données et l'intégration des données dans l'industrie du pétrole

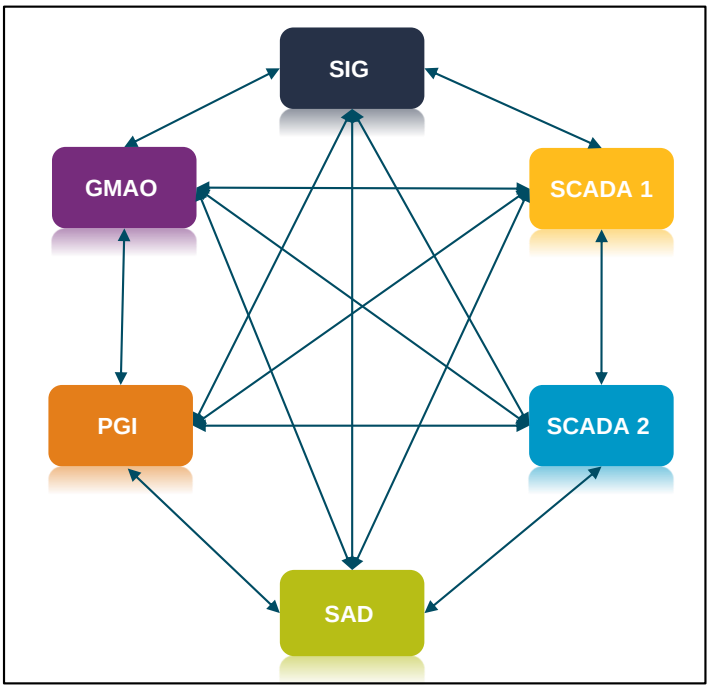


FIGURE 2.10 – Les solutions ad-hoc, (adapté de [Izza, 2009])

et du gaz [Council, 2008]. Il y a ISA-95 pour Enterprise-Control System Integration [Instrumentation and Society, 2010], MIMOSA pour les solutions d'interopérabilité du pétrole et du gaz [12], ISO-15926 pour la modélisation des données et l'intégration des données pour l'industrie du pétrole et du gaz [Council, 2008], ou PRODML [13] dans l'industrie du pétrole et du gaz; ISO-10303 [Pratt, 2001b] dans les systèmes d'automatisation et l'échange de données produit. Pour les systèmes qui ne sont pas interopérables avec une norme commune, une solution ad hoc doit être utilisée pour mapper ces systèmes à la norme. Cette approche est référée à la normalisation de l'interface (Interface Standardization) [Wegner, 1996].

2.2.3 Les cadres d'architectures

Tous les aspects d'interopérabilité abordés précédemment doivent être appliqués dans une solution architecturale afin que tous les aspects soient collés et se complètent. Par conséquent, certains groupes de travail et organisations se mobilisent pour décomposer les problèmes d'interopérabilité en définissant un modèle de référence dans les approches architecturales. Ces modèles de références aident à définir une feuille de route pour coller les aspects dans un environnement particulier. Grid Wise Architectural Council (SGAM) [Council, 2008] et Smart Grid Architecture Model (SGAM) [CEN-CENELEC-ETSI and Coordination, 2012] sont des modèles d'architecture de référence pour le domaine de l'électricité. En Al-

[12] http ://www.mimosa.org/
[13] http ://www.energistics.org/production/prodml-standards

lemagne, le Deutsches Institut für Normung (DIN) et d'autres organisations ont publié la "Reference Architecture Model for Industry 4.0 (RAMI 4.0)" pour la quatrième révolution industrielle [Hankel and Rexroth, 2015]. En Chine, l'Administration de normalisation de chine (SAC) et le Ministère de l'Industrie et de l'Information Technologie (MIIT) publie la "National Smart Manufacturing Standards Architecture Construction Guidance".

2.2.3.1 L'architecture "Smart Grid Architecture Model (SGAM)"

Le Smart Grid Architecture Model (SGAM) est un cadre de référence défini par le groupe de travail "EU Mandate M/490's Reference Architecture" pour représenter les architectures Smart Grid [CEN and ETSI, 2012a]. Il s'agit d'un modèle tridimensionnel (cf. figure 2.11) qui fusionne la dimension de cinq couches d'interopérabilité avec les deux dimensions de Smart Grid Plane (les zones et les domaines). Le SGAM permet d'établir des relations entre les couches abstraites de problèmes d'interopérabilité (Business, Fonctions, Information, Communication, Composant) et les présente d'un manière neutre sur le plan technologique. La Couche de Business représente le point de vue de l'entreprise sur l'échange d'informations liées aux réseaux intelligents tels que les objectifs et processus commerciaux, entités organisationnelles, conditions réglementaires, les politiques. Par exemple, la procédure à suivre quand il y a des problèmes dans certains points d'injection, le comportement du système quand le prix s'augmente, etc. La couche de fonctions décrit les fonctions et les services, y compris leurs relations à partir d'un point de vue architectural pour réaliser les aspects commerciaux respectifs. La couche d'Information décrit l'information qui est utilisée et échangée entre fonctions, services et composants. Elle représente aussi la sémantique pour l'échange d'informations en utilisant des objets et des modèles de données. La couche de communication décrit les protocoles et les mécanismes d'échange d'informations interopérables entre les composants en définissant la syntaxe. Enfin, la couche de composants met l'accent sur tous les composants qui sont interconnectés au système tel que les capteurs, actuateurs, les applications, les logiciels, etc.

Le Smart Grid Plane couvre d'une part les domaines de la chaîne de conversion d'énergie (Generation, Transmission, Distribution, Distributed Energy Resources (DER) and Customer Premises). D'un autre part, il couvre les zones hiérarchiques de la gestion du système d'alimentation (Process, Field, Station, Operation, Enterprise, Market).

Il a pour but de présenter la conception des cas d'utilisation de SmartGrid dans un point de vue architectural. Il présente les couches d'interopérabilité d'une manière neutre sur le plan technologique. En fait, il donne un cadre de modélisation qui prend en compte l'interopérabilité, mais il ne donne pas les détails techniques pour gérer cette interopérabilité. Il est utilisé par les organismes de normalisation pour identifier les lacunes de normalisation [CEN and ETSI, 2012b].

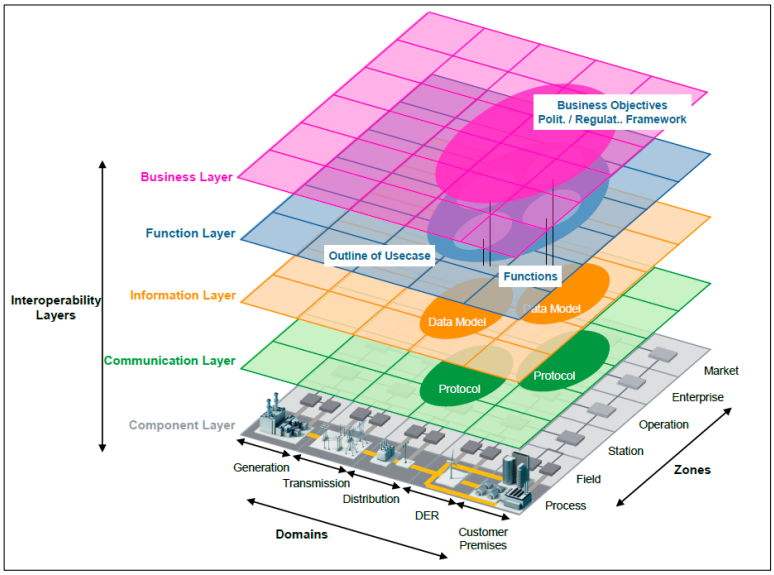


FIGURE 2.11 – Le cadre SGAM [CEN and ETSI, 2012a]

2.2.3.2 L'architecture "Reference Architecture Model for Industry 4.0 (RAMI 4.0)"

Cette architecture [Peter Adolphs, 2015], qui est basée sur l'architecture SGAM, décrit les aspects essentiels pour l'industrie 4.0. C'est un modèle tridimensionnel (cf. figure 2.12) pour lier les informations (IT), les fabricants, les usines et le cycle de vie du produit : Hierarchy Levels, Life Cycle and Value Stream et Layers.

L'axe horizontal droit, "Hierarchy levels", représente l'emplacement des fonctionnalités et des responsabilités dans les usines. Cela représente une hiérarchie fonctionnelle de produits, capteurs et actionneur (Field device) jusqu'à l'entreprise et le monde connecté (connected world). Traditionnellement, ces niveaux sont considérés comme une "véritable hiérarchie" et représentés comme une pyramide, cependant, avec l'industrie 4.0, ils sont plus conçus et représentés comme un maillage dans une réalité de connectivité omniprésente (ubiquitous connectivity en anglais) de tout.

L'axe horizontal gauche, "Life cycle and value stream", représente la modélisation de données tout au long du cycle de vie du produit. Il distingue deux aspects : "Type" et "Instance". L'aspect "Type" décrit le produit pour commencer, tester et valider la fabrication. L'aspect "Instance" décrit le produit après la fabrication, pendant l'utilisation et jusqu'à la disposition.

L'axe vertical gauche, "Layers", définit la structure de la représentation informatique d'une machine ou d'un objet. Il représente (i) l'entreprise et ses processus d'affaires, (ii) les fonctions requises pour les actifs (iii) les données échangées (iv) communication pour accéder à l'information (v) les capacités d'intégration, c.-à-d. transition du monde physique au monde numérique, (vi) les actifs(physical assets).

La combinaison de ces trois axes résulte dans une architecture 3D orientée services qui fournit les bases pour implémenter Industrie 4.0. RAMI fournit une structure et un langage communs pour la description et la spécification uniformes des architectures du système. Il sert à une compréhension commune, quelles normes, quel cas d'utilisation sont nécessaires pour l'Industrie 4.0. L'architecture permet l'identification des normes, les écarts entre eux et les liens entre eux.

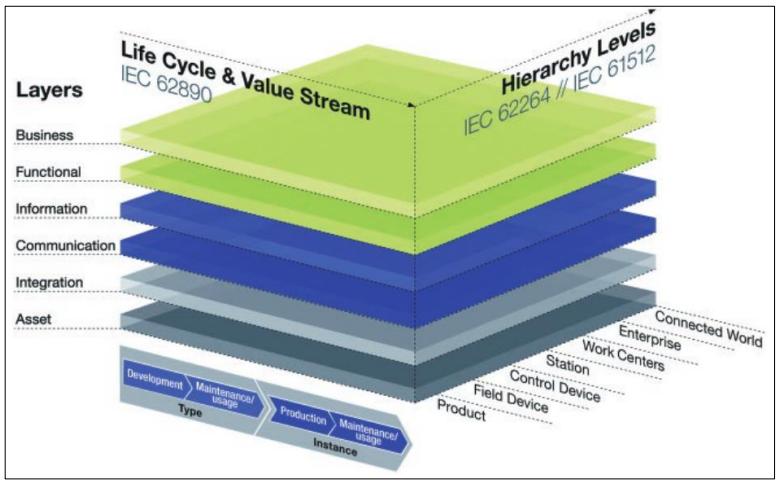


FIGURE 2.12 – L'architecture RAMI [Peter Adolphs, 2015]

2.2.3.3 L'architecture "Industrial Internet Reference Architecture"

L'architecture Industrial Internet Reference Architecture (IIRA) [Shi-Wan Lin and Witten, 2015] est une architecture normée pour connecter des systèmes de contrôle industriel à des personnes et les intégrer pleinement aux systèmes d'entreprise, aux processus métier et aux solutions analytiques en créant ainsi un système de bout en bout. Il fournit des modèles d'architecture et une méthodologie pour aider les architectes système IIdO à concevoir des architectures de solution IIdO de manière cohérente et à déployer des systèmes IIdO interopérables basés sur un cadre et des concepts communs.

L'IIRA est décrit en utilisant quatre points de vue : business, usage, functional, et implementation (cf. figure 2.13). Chaque point de vue est constitué de conventions encadrant la description et l'analyse de préoccupations spécifiques du système. En plus, chaque point de vue est associé aux parties prenantes qui sont des individus, des équipes, des organisations ayant un intérêt dans un système.

Le Point de vue "Business " porte sur les préoccupations axées sur les entreprises telles que la vision, les valeurs et les objectifs d'affaires. Ces préoccupations sont identifiées par les parties prenantes qui sont principalement les décideurs d'affaires, les chefs de produits et les ingénieurs systèmes.

Le point de vue "Usage" répond aux préoccupations liées à l'utilisation et les capacités du système identifiées dans le point de vue "Bussiness". Il est représenté comme des séquences d'activités impliquant des utilisateurs humains ou logiques qui décrivent comment un système est utilisé. Ces préoccupations présentent un intérêt particulier pour les ingénieurs systèmes, les chefs de produit et les analyseurs de systèmes impliqués dans la spécification du système.

Le point de vue "Functional" se concentre sur les composants fonctionnels, leur structure, les interfaces et les interactions entre eux et la relation et les interactions du système avec les éléments externes de l'environnement pour soutenir les usages et les activités de l'ensemble du système. Ces préoccupations intéressent particulièrement les architectes, développeurs et intégrateurs de systèmes et de composants.

Le point de vue "Implementation" décrit les technologies nécessaires à la mise en œuvre des activités et des fonctions prescrites par les points de vue "Usage" et "Functional". Ces préoccupations intéressent particulièrement les architectes de systèmes et de composants, les développeurs et les intégrateurs, ainsi que les opérateurs du système.

Cette architecture fournit un cadre pour piloter les projets IIdO d'un point de vue du métier. Les fournisseurs de technologie peuvent utiliser les concepts et les méthodologies de l'IIRA pour construire des composants de système interopérable qui s'adressent à un grand marché. Les développeurs peuvent utiliser l'IIRA comme point de départ pour réduire le développement du système en déployant des blocs de système réutilisables et des solutions open source afin de réduire les coûts de développement et le délai de mise sur le marché. Enfin, l'IIRA aidera la communauté de l'IIdO à réaliser un écosystème IIdO ouvert et innovant en réduisant le coût de la conception.

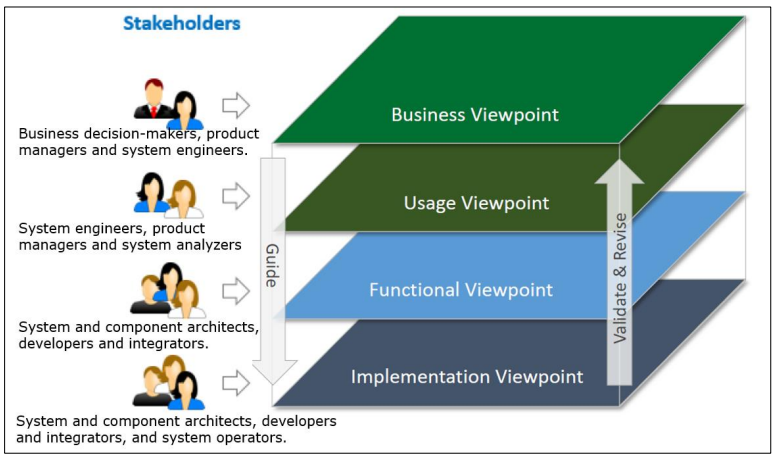


FIGURE 2.13 – L'architecture IIRA

2.2.3.4 IOT Architectural Reference Model (ARM)

L'IOT Architecutre (IOT-A) [Bassi et al., 2013, Martin Bauer, 2013] a créé un modèle de référence architectural (Architectural Reference Model - ARM) comme un terrain commun pour l'IdO. L'IdO ARM (cf. figure 2.14) est utilisé pour générer des architectures IdO concrètes adaptées à des applications spécifiques et à des cas d'utilisation. Il est basé sur un modèle de référence (Reference Model) qui fournit l'abstraction la plus élevée pour promouvoir une compréhension commune et une langue commune du domaine cible de l'IdO. Cette compréhension est basée sur les scénarios commerciaux actuels, les architectures existantes et les parties prenantes. Ensuite, le modèle de référence utilise quelques lignes directrices pour générer une architecture de référence (Reference Architecture).

Cette architecture de référence fournit des vues architecturales et les perspectives au niveau abstrait qui sont pertinentes pour les systèmes IdO. D'une part, une vue est une représentation d'un ou plusieurs aspects structurels et préoccupations d'une architecture comme la vue fonctionnelle, la vue d'information, etc. D'autre part, une perspective architecturale est un ensemble d'activités et de lignes directrices qui permettent de s'assurer qu'un système présente un ensemble particulier de propriétés de qualité qui nécessitent une prise en compte dans plusieurs points de vue comme la sécurité, la disponibilité, l'interopérabilité, etc. Enfin, cette architecture de référence utilise aussi les lignes directrices pour créer l'architecture concrète. Cette architecture est très abstraite et fournit une approche pour modéliser le système IdO et générer l'architecture concrète qui doit être mis en œuvre. Cependant, il ne donne pas de détails techniques pour mettre en ouvre cette architecture concrète

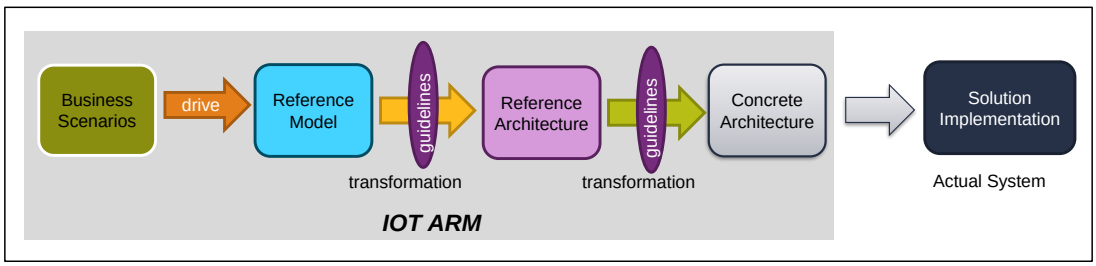


FIGURE 2.14 – L'architecture IOT ARM

2.2.3.5 Les Architectures Orientées Services

Les Architectures orientées services, Service Oriented Architecture (SOA) en anglais, est une architecture métier conceptuel où les fonctionnalités métiers, ou la logique de l'application, est mise à disposition aux utilisateurs SOA ou aux utilisateurs, en tant que services réutilisables et partagés dans un environnement informatique. Les services dans une SOA sont les modules d'une fonctionnalité de l'application avec des interfaces exposées qui sont invo-

quées par messages [Marks and Bell, 2006]. Le type de service le plus largement accepté est le service Web (ou service Web XML). Il communique via des protocoles comme HTTP/SOAP et HTTP Rest et envoie les données en format tel que XML et JSON. Le système producteur fournit les services qui sont accessibles via le réseau pour d'autres systèmes consommateurs de ce service. Ces services et leurs consommateurs correspondants communiquent entre eux en transmettant des données dans un format partagé bien défini ou en coordonnant une activité entre deux ou plusieurs serveurs. Les technologies de services Web ont évolué pour prendre en charge des fonctionnalités plus avancées telles que la découverte, la sécurité, les transactions, la fiabilité et la gestion des processus collaboratifs. Plus de détails de ces architectures sont disponibles dans [Erl, 2005, Marks and Bell, 2006].

2.2.3.6 OPC Unified Architrecutre

OPC (OLE(**O**bject Linking and Embedding) for **P**rocess **C**ontrol) est une norme pour l'échange d'informations utilisée dans l'environnement industriel. Elle est maintenue et développée par OPC Foundation [14]. Sa spécification initiale OPC est connue par "OPC Classic". Le successeur et la dernière spécification OPC s'appelle OPC UA (**U**nified **A**rchitecture) [Mahnke et al., 2009]. Contrairement à l'ancienne spécification qui ne prend en charge que les mécanismes de transport, cette dernière spécification est une norme et une architecture qui prend en charge à la fois les mécanismes de transport, la modélisation des données et les services. Nous avions déjà parlé d'OPC UA pour l'interopérabilité technique et sémantique. Cepandant, OPC UAest une architecte qui fournit plus de fonctionnalités en termes d'indépendance de plate-forme, d'évolutivité et de sécurité.

OPC UA est à la fois une norme de communication, de modélisation de données, et une architecture qui se compose de 13 parties réparties en trois groupes, comme indiqué dans la figure 2.15. OPC UA est une architecture client-serveur ce qui signifie que vous avez un ou plusieurs serveurs qui attendent plusieurs clients pour faire des demandes. En outre, le client peut également s'abonner pour recevoir des mises à jour du serveur. OPC UA fournit tous les éléments comme les SDK, Stack, guides pour réaliser les solutions concrètes OPC UA. Il existe aussi quelques logiciels clients OPC UA tels que UA Expert (solution gratuite) [15] et des solutions commercialisées telles que Panorama [16] et Wonderware [17] afin d'interagir avec les serveurs OPC UA.

En fait, OPC UA est considéré comme une technologie habilitante pour l'industrie 4.0 [Imtiaz and Jasperneite, 2013]. D'une part, il définit comment l'information entre les partenaires de communication est transférée. D'autre part, il spécifie les sémantiques via des modèles d'informations agnostiques de domaine qui peuvent être entendus avec des mo-

[14] https ://opcfoundation.org/
[15] https ://www.unified-automation.com/products/development-tools/uaexpert.html
[16] https ://www.codra.net
[17] https ://www.wonderware.com/

dèles spécifiques à l'utilisateur. Dans ce mode hétérogène, les systèmes doivent être OPC UA-activé afin de promouvoir l'interopérabilité entre eux sinon il faut utiliser les adaptateurs ad-hoc ou des outils commercialisés comme keepware [18].

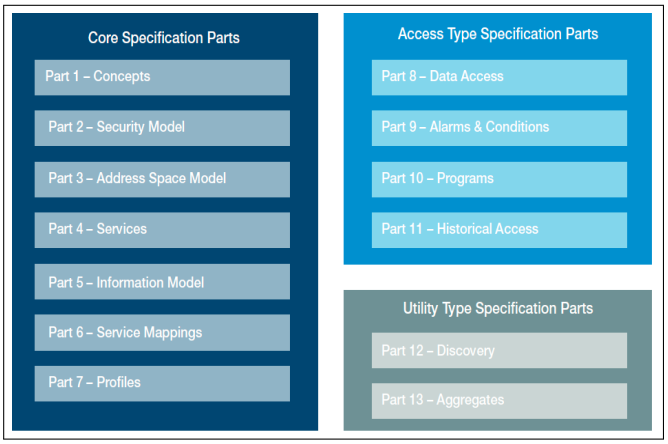


FIGURE 2.15 – OPC UA specifications

2.2.4 Les middlewares

Middleware, qui est un cadre convenu d'un commun accord, se pose pour intégrer des systèmes hétérogènes, par exemple, pour intégrer une base de données, un site Web front-end (un frontal) et une application principale (un arrière-plan). Middleware, qui est considéré comme de la colle de logiciel, gère l'interaction entre des applications disparates sur les plates-formes informatiques hétérogènes. Il y a plusieurs définitions pour les middlewares et nous sommes intéressés par la définition donnée par le centre de ressources [Misra, 2011]. Il définit Middleware comme tout logiciel permettant à d'autres logiciels d'interagir. Diverses approches de conception existent pour les solutions de middleware [Izza, 2009, Razzaque et al., 2016b, Serain, 2002], telles que Middleware basé sur les appels de procédures à distance, Middleware basé sur les événements, Middleware orienté service, Middleware orienté base de données, Middleware basé sur les agents, Middleware orientés application, Middleware basé sur les machines virtuelles, etc. Dans la section suivante, nous présentons une brève présentation de certaines approches sans entrer dans les détails techniques.

2.2.4.1 Middleware basé sur les appels de procédures à distance

Cette approche s'appelle "Remote procedure call - RPC" middlewares. Il est basé sur les appels de procédures distants à travers le réseau. Ce mécanisme est utilisé dans le modèle client/serveur généralement mis en œuvre via un système de transmission Request/Reply : un client se connecte à un serveur distant et invoque les services qui sont fournis par le ser-

[18] https ://www.kepware.com

veur. Dans le paradigme de programmation orientée objet, les appels RPC sont représentés par l'invocation de méthodes distantes (Remote method invocation - RMI en anglais). Ces manipulations induisent des traitements et des modifications d'informations dans l'application qui est propriétaire de l'objet en question. Il y a plusieurs environnements à base de RPC, par exemple : Distributed Computing Environment (DCE) RPC [Shirley, 1992] fournit un cadre et des outils pour développer des applications client-serveur. Cependant, il ne prend pas en charge le langage orienté objets ce qui le rend moins utile aujourd'hui.

Le Java RMI (Remote method invocation) est dédié pour les clients et serveurs java. Avec Java RMI, les méthodes d'un objet distant java peuvent être invoquées par un autre objet java [Bernstein, 1996, Wollrath et al., 1996].

Distributed Component Object Model (DCOM) de Microsoft est aussi une technique de manipulation d'objets distants. Il est utilisé pour des protocoles de communications comme OPC UA. Cependant, DCOM soutient un nombre restreint des systèmes d'exploitation, plus précisément Windows.

CORBA (Common Object Request Broker Architecture) est une architecture orientée objet qui permet aux objets d'interopérer à travers les réseaux indépendamment de la langue sur laquelle ils sont écrits ou la plate-forme sur laquelle ils sont déployés [He and Xu, 2014, OMG, 2012] ; etc.

2.2.4.2 Middleware basé sur les événement

Dans cette approche, appelée aussi Event-Based Middlewares en anglais, tous les systèmes et applications interagissent à travers les événements. Un système produit un événement qui est consommé par un autre système [Meier and Cahill, 2002b]. Ils sont principalement basés sur le modèle de message Publish/subscribe, par conséquent, il est difficile d'obtenir un modèle de demande / réponse avec une telle approche [Neely et al., 2006]. Les consommateurs, qui sont enregistrés pour des événements particuliers, s'abonnent aux fournisseurs de flux de données via un référentiel central. Lorsqu'un événement est déclenché, le consommateur de données reçoit les données. Le middleware orienté messages (MOM), Message-oriented middleware en anglais, est un type de middleware basé sur des événements où l'interaction est basée sur des messages. En effet, ces messages sont des événements avec des métadonnées supplémentaires. Les exemples de middleware basés sur les événements incluent : EMMA [Musolesi et al., 2006] qui est utilisé dans des environnements ad-hoc mobiles ; GREEN [Sivaharan et al., 2005] qui est utilisé dans les applications informatiques omniprésentes ; Hermes [Pietzuch, 2004] qui est utilisé dans les systèmes distribués à grande échelle ; RUNES [Costa et al., 2007] qui est utilisé entre un réseau hétérogène de systèmes embarqués ; Steam [Meier and Cahill, 2002a] dans le domaine informatique mobile ; PSWare [Lai et al., 2009], Mires [Souto et al., 2006], SensorBus [Ribeiro et al., 2005] sont des middlewares pour les réseaux de capteurs sans fil (Wireless sensor network - WSN en anglais) ; TinyDDS [Boonma and Suzuki, 2010], qui est basé sur la norme Data Distribution Services,

(DDS), est utilisé pour l'interaction entre WSN et les réseaux d'accès. En effet, la plupart de ces middlewares traitent de l'interopérabilité au niveau technique sans tenir compte de l'interopérabilité syntaxique et sémantique.

2.2.4.3 Middleware orientés service

Ces middlewares, appelé Service-Oriented Middlewares - SOM en anglais, facilitent la conception et le développement de solutions SOA (Service Oriented Architecture) concrètes. La SOA permet de partitionner une application en un ensemble de services. Un service est une unité discrète de fonctionnalité qui peut être consultée à distance, exploitée et mise à jour indépendamment [Lavoie et al., 2006], par exemple la récupération d'un relevé de carte de crédit en ligne. Ainsi, SOM construit des logiciels et des applications sous forme de services.

2.2.4.4 Middleware orienté base de données

Ces middlewares, appelés Database-Oriented Middleware - DOM en anglais, interopèrent des systèmes en établissant une base de données virtuelle commune. Il permet la communication d'applications à différents types de bases de données en utilisant une interface unique, quels que soient le modèle utilisé ou les plates-formes sur lesquelles elles existent. Par exemple (cf. Figure 2.16), DOM utilise l'interface "Call Level Interface (CLI)" telle que les APIs ODBC (Open Database Connectivity) ou JDBC (Java Database Connectivity) pour accéder aux informations dans différentes bases de données comme Oracle, Informix, DB2, etc. Des langages de requête sont utilisés pour interroger les bases de données comme SQL, ce qui permet la formulation de requêtes complexes. Dans le monde IdO, les choses sont vues comme des bases de données virtuelles qui sont interrogées par des applications. Exemples de DOM pour WSN incluent COUGAR [Bonnet et al., 2001], IrisNet [Gibbons et al., 2003], Sensation [Hasiotis et al., 2005], KSpot+ [Andreou, 2011], etc. Cette approche considère l'ensemble du réseau comme un système de base de données virtuel. Par conséquent, tous les dispositifs ou systèmes fournissent une interface accessible pour interroger leur base de données, ce qui n'est pas possible dans un monde hétérogène. De plus, les DOM ne sont pas efficaces dans l'application en temps réel, ce qui est un besoin essentiel dans les systèmes IdO actuels [Razzaque et al., 2016a].

2.2.5 Solutions pour les environnements hétérogènes

De nombreuses plate-formes existent pour gérer l'interopérabilité dans des environnements hétérogènes : IBM Watson IoT Platform [19], Azure IoT Platform [Azu, 2016], Oracle IOT cloud [20], COOperate [Ellis et al., 2017], etc. la plupart des solutions sont basées SOA et SOM. En fait,

[19] https ://www.ibm.com/internet-of-things
[20] https ://cloud.oracle.com/iot

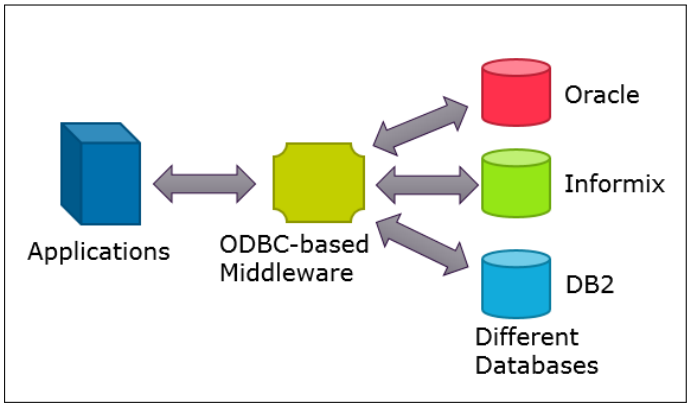


FIGURE 2.16 – Un exemple de DOM pour accéder à diverses bases de données

une enquête est donnée par [Ray, 2016]. Dans ce rapport, nous présentons techniquement certains d'entre eux.

2.2.5.1 IBM Watson IoT

La plate-forme IBM Watson IoT [21] fournit aux applications un accès puissant aux dispositifs, terminaux et aux données IoT. De plus, il permet de composer des applications d'analyse, des tableaux de bord de visualisation et des applications IoT mobiles. Cependant, la plate-forme est limitée à un ensemble spécifique de protocoles de communication tels que MQTT et HTTP. De plus, il fournit une API REST pour communiquer entre les applications et les dispositifs. Au niveau syntaxique, la plate-forme utilise JSON comme format d'échange et la définition de structure de données. Au niveau sémantiques, la plate-forme permet de définir les modèles de données spécifiques au types de dispositifs. Si plusieurs systèmes partagent les mêmes sémantiques, la plate-forme permet de créer une représentation digitale appelée un terminal jumeau. Un terminal jumeau crée un modèle logique des propriétés et des événements provenant d'un capteur ou d'un terminal spécifique. Ensuite, pour chaque terminal physique connecté, il faut ajouter les règles de mapping entre le modèle de données de terminal connecté et le modèle logique. Le mapping se base sur JSONata [22] qui est un langage de transformation de données JSON. Ce mapping peut impliquer un traitement de données tel qu'une conversion d'unité, multiplier par une constante, etc. L'application IoT peut accéder aux données du terminal selon leur définition par modèle logique sans tenir compte du modèle de contenu du message de terminal.

Par exemple, nous avons deux capteurs de température qui communique via la plate-forme comme illustré dans la figure 2.17 [23] . Le capteur de température 1 publie sur Watson IoT Platform une mesure de température en degrés Celsius "t" : 34.5 . Le capteur de tempé-

[21] https ://www.ibm.com/internet-of-things
[22] http ://jsonata.org/
[23] https ://console.bluemix.net/docs/services/IoT/GA_information_management/ga_im_definitions.html

rature 2 publie une mesure en degrés Fahrenheit "temp" : 72.55 . Chaque capteur de température est associé à son propre type de terminal. Les données "t" et "temp" sont mappées vers un modèle logique définissant la mesure température. Enfin, l'application accède aux données de capteur selon le modèle logique. Pour résumer, cette plate-forme permet aux applications d'accéder aux données des terminaux, capteurs via des protocoles spécifiques MQTT/HTTP et des API Rest. Il gère la sémantique au niveau de dispositif et pas au niveau des applications.

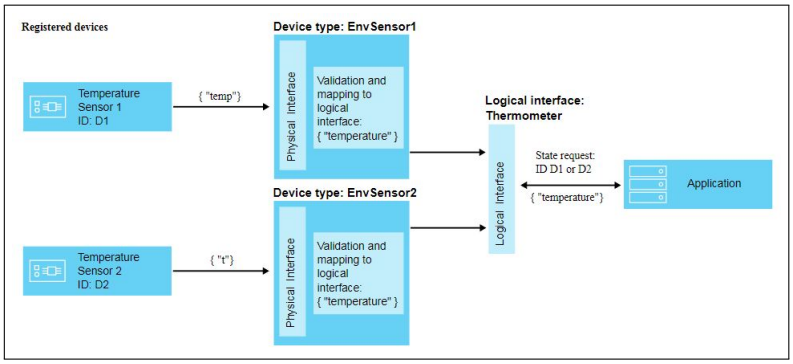


FIGURE 2.17 – Un exemple de capteurs via le plate-forme IBM Watson IoT

Pour résumer, IBMWatson est une plate-forme générique. Elle prend en charge quelques normes/protocoles de communications et formats. Elle permet de définir les sémantiques de dispositifs et leurs règles de mapping. Cependant, il n'est pas extensible pour d'autres normes/protocoles. Il faut avoir des adaptateurs externes pour interoperer avec les systèmes non compatibles en termes de normes, protocoles, sémantiques avec la plate-forme. De plus, cette plate-forme gère seulement l'échange vertical de données (niveau OT). Les applications consommatrices doivent être compatibles avec les protocoles MQTT/HTTP et les API Rest pour les interoperer avec les données agrégées des dispositifs. En outre la plate-forme ne permet pas de définir les sémantiques spécifiques avec les règles métiers/opérationnelles et leurs applications.

2.2.5.2 Azure IoT

La plate-forme Azure IoT [Azu, 2016], qui gère l'intégration de millions de terminaux avec un système basé sur le cloud dans le but de l'analyse, le contrôle et l'intégration des processus d'affaires, propose une architecture abstraite pour réaliser des solutions concrètes. Cette plate-forme est générique et orientée service. Elle permet de gérer les terminaux, les historiques de données et l'intégration avec les applications de l'entreprise. Elle fournit de nombreuses fonctionnalités, mais nous l'examinons du point de vue des niveaux d'interopérabilité. La plate-forme Azure a un composant Azure IOT hub qui permet de communiquer avec les terminaux. Il prend en charge les protocoles AMQP, MQTT et HTTP et prend en charge des protocoles supplémentaires en utilisant un modèle d'adaptation ou d'API de Hub. Les

applications au niveau IT sont intégrées avec des connecteurs ou API. Au niveau syntaxique, la plate-forme utilise les formats JSON comme format neutre. Cependant, au niveau sémantique, il est supposé que les appareils, applications et composants de la plate-forme sont compatibles.

2.2.5.3 Oracle IOT cloud

Une autre plate-forme pour permettre l'interopérabilité est Oracle IOT cloud footnotehttps ://cloud.oracle.com/iot. Cette plate-forme, qui est générique au niveau de domaine et orientée services, permet de connecter des terminaux existants à des moteurs d'analyses et de business intelligence dans le cloud. Cette plate-forme utilise un bus de service (Service Bus) pour interagir avec les applications et les terminaux. Les protocoles pris en charge sont Oracle HTTP/REST, HTTP/SOAP, WS-I et JMS. Les formats de données sont JSON. C'est le rôle de l'application et des dispositifs d'interpréter la sémantique des données ou de la mapper à ses sémantiques. Cependant, la plupart de ces outils se concentrent sur l'analyse de données en prenant en compte l'interopérabilité des silos verticaux. De plus, ces solutions sont basées sur des services, c'est l'application ou les terminaux qui gèrent les sémantiques.

2.2.5.4 SOCRADES

L'architecture d'intégration SOCRADES pour l'IdO est une middleware SOM qui permet l'intégration entre des dispositifs (objets) et des applications en utilisant les services web [Spiess et al., 2009]. Cette architecture permet l'intégration de dispositifs hétérogènes via des plug-ins qui convertissent et représentent les données de dispositifs comme les services. Cependant, les applications sont orientées service et l'architecture ne gère pas l'interopérabilité au niveau sémantique. C'est le rôle de l'application ou les dispositifs d'interpréter les sémantiques de données. L'architecture prend en compte l'interopérabilité des silos verticaux d'échange de données. Pour résumer, cette architecture gère l'interopérabilité technique et syntaxique via plug-ins au niveau dispositifs et via une approche d'intégration au niveau d'application en imposant que les applications soient orientées service.

2.2.5.5 HYDRA

HYDRA [Eisenhauer et al., 2010] est une SOM pour les systèmes embarqués en réseau qui permettent aux développeurs de créer des applications d'intelligence ambiante (AmI), c'est-à-dire des environnements électroniques sensibles et réactifs à la présence de personnes, basées sur des dispositifs sans fil et des capteurs. Il permet aux développeurs d'intégrer des dispositifs physiques hétérogènes avec leurs applications. D'une part, il gère le niveau technique et syntaxique en utilisant les service web SOA. D'autre part, il gère l'interopérabilité sémantique en utilisant des ontologies au niveau des dispositifs. HYDRA permettra le développement de services génériques basés sur des standards ouverts. Ce middleware se concentre

sur l'interopérabilité des silos verticaux d'échange de données. Cependant, tous les dispositifs et les applications doivent être basés service. Ces middlewares se concentrent également sur les aspects de performance tels que la découverte, la QoS, le stockage, etc.

2.2.5.6 COOPERaTe

Le projet COOPERaTe [Ellis et al., 2017] vise à agréger, à analyser et à agir sur les données issues de plusieurs systèmes dans des quartiers à énergie positive EPN (Energy Positive Neighborhoods en anglais) via une approche SOA. EPN est un concept au niveau du système où le voisinage génère plus d'énergie qu'il n'en consomme, le surplus d'énergie étant stocké localement ou exporté. Dans ce contexte, le projet a proposé la plate-forme/architecture COOPERaTe qui aborde l'interopérabilité pour l'échange de données de bout en bout entre les systèmes hétérogènes. Cette plate-forme est orientée SOA et fournit les services pour les systèmes dans les quartiers. Il permet d'étendre la plate-forme pour gérer l'interopérabilité technique. Au niveau de l'interopérabilité sémantique, il utilise le métamodèle NIM "Neighborhood Information Model (NIM)" [Greifenberg et al., 2014] en tant que métamodèle convenu entre les systèmes hétérogènes. Donc, si les systèmes ne conviennent pas au NIM, il faut toujours les adapter via l'API avant de les utiliser.

2.2.5.7 Xively

Son objectif principal est de connecter les appareils différents au système de cloud, où vous pouvez facilement extraire des données de n'importe où et n'importe quand. Cependant, les communications de la plate-forme avec des dispositifs et les applications sont basées sur le protocole de communication MQTT ou sur une API qui prend en charge REST, WebSockets et MQTT. II ne prend en compte que certains formats tels que JSON, XML et CSV. Il est le rôle des consommateurs d'interpréter les données qui peuvent être accessibles de partout à partir du cloud via l'API Xively. En fait, cette application se concentre sur l'acquisition des données des appareils et les rend disponibles dans le système cloud pour un traitement ultérieur plutôt que de gérer l'interopérabilité. Autrement dit, cette plate-forme se concentre sur l'interopérabilité technique et syntaxique des silos verticaux pour un nombre limité de protocoles et de formats. Il ne fournit pas de solution pour gérer l'interopérabilité au niveau sémantique et les autres niveaux d'interopérabilité pour les appareils non compatibles avec celui-ci.

2.2.5.8 Arrowhead

Le projet Arrowhead cible des domaines d'activité telles que la production, les bâtiments intelligents et leurs infrastructures, l'électromobilité, la production d'énergie et les marchés virtuels de l'énergie afin de permettre l'interopérabilité entre les systèmes hétérogènes. Il

propose un cadre SOA-basé intitulé Arrowhead. Son objectif est de permettre l'interopérabilité et l'intégration pour l'IdO, l'IdTO industriel et le système de systèmes [Varga et al., 2017]. Il se concentre sur l'interopérabilité des applications SOA en termes de : i) comment un système fournisseur de services fait-il connaître ses services au consommateur ? ii) comment un consommateur de services découvre-t-il les services qu'il veut consommer ? iii) Comment un fournisseur de services décide-t-il si un système qui veut consommer ses services est autorisé à le faire ? iv) comment orchestrer le système de systèmes, c'est-à-dire permettre à un organe d'orchestration de contrôler quel service un système doit consommer ? Le framework est composé d'un certain nombre de services de base tels que le registre de service, l'orchestration et l'autorisation pour interagir avec d'autres services d'application (cf. figure 2.18). Pour qu'une application soit conforme à Arrowhead, elle doit enregistrer les services qu'elle produit avec un registre de service Arrowhead. Elle utilise ensuite le service d'autorisation Arrowhead pour accepter ou refuser les applications consommatrices. Ensuite, le système d'orchestration permet la reconfiguration dynamique du consommateur et du fournisseur de services.

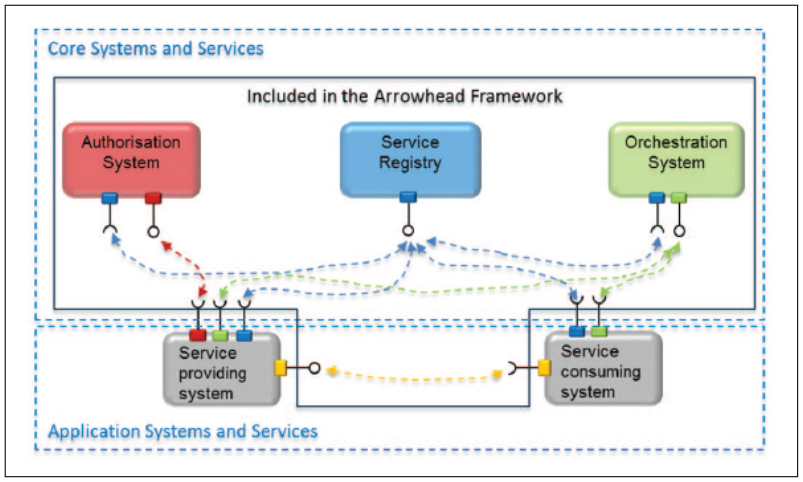


FIGURE 2.18 – Le noyau du framework Arrowhead et les services d'application

Le framework Arrowhead prescrit également une méthode de documentation afin de permettre une interopérabilité native entre ces services principaux et les services fournis/consommés par l'application compatible Arrowhead. Cependant, afin de rendre les services transparents pour les systèmes basés sur des sémantiques et des protocoles de communication différents, le framework Arrowhead suggère l'utilisation d'une couche Interoperability/Translator. Cette couche utilise des adaptateurs ad-hoc pour convertir les applications non-Arrowhead en conformité avec Arrowhead et ainsi assurer l'interopérabilité avec le framework [Varga et al., 2017].

2.2.5.9 OpenIoT

le projet [Soldatos et al., 2015] a fourni une plate-forme de middleware qui permet d'intégrer des dispositifs avec les applications au niveau d'échange de données vertical OT. Cette solutions permet de communiquer avec des dispositifs hétérogènes via une middleware dédié à la communication. Ensuite, il utilise l'ontologie des réseaux de capteurs sémantiques (SSN) du W3C comme un modèle basé sur des standards communs pour l'unification sémantique de divers systèmes IdO pour gérer la sémantique. La plate-forme permet de définir les services sur les données afin de les visualiser par les applications de visualisation. Donc, ce projet est orienté vers le domaine de capteurs et il ne permet que d'integer les applications au niveau de OT et pas IT.

2.2.5.10 Solutions divers

[Blackstock and Lea, 2012, Maureira et al., 2011] sont des hubs pour l'IdO qui regroupent, traitent et visualisent des informations à partir d'objets physiques et les exposent sur le web. Ces solutions sont des SOM qui se concentrent sur l'interopérabilité des silos verticaux d'échange de données. MUSIC [Rouvoy et al., 2009] est une autre SOM qui facilite la construction de systèmes capables de s'adapter dynamiquement à un contexte d'exécution variable chez les fournisseurs de services et les consommateurs de services. TinySOA [Avilés-López and García-Macías, 2009] est un SOM pour le développement d'applications WSN. D'autres exemples de SOM incluent SensorsMW [Anastasi et al., 2010], ubiSOAP [Caporuscio et al., 2012], KASOM [Corredor et al., 2012], Servilla [Fok et al., 2012], SenseWrap [Evensen and Meling, 2009], MOSDEN [Perera et al., 2014], etc. Toutes ses solutions sont des solutions orientées service. Elles ne sont pas suffisantes dans notre contexte parce qu'il faut interagis avec les systèmes hétérogènes qui ne sont pas tous orientés services.

2.2.6 Synthèse

Dans la section 2.1, nous avons présenté un certain nombre d'approches qui traitent de l'interopérabilité à leurs différents niveaux. Ensuite, dans cette section, nous avons présenté les solutions/techniques ad-hoc, normes, architectures, middlewares qui servent à coller et à traiter plus d'un niveau d'interopérabilité comme illustré dans la figure 2.19.

Les solutions ad-hoc sont très coûteuses et complexes à gérer, car les solutions sont non extensibles, non modulaires et dépendantes du système ou application. De plus, le nombre de mapping augmente avec le nombre de systèmes intégrés dans l'environnement. Les impositions de normes, par exemple, [Council, 2008, Instrumentation and Society, 2010], MIMOSA, PRODML pour l'industrie gazière et pétrolière. Cependant, les auteurs de [Veyber et al., 2012] ont conclu qu'il n'existe pas de norme unifiée pour l'intégration des systèmes dans l'industrie pétrolière et gazière. D'autres travaux dans [Veyber et al., 2010] ont également montré les limites de ces normes pour les réseaux de gaz intelligents. L'Institut national

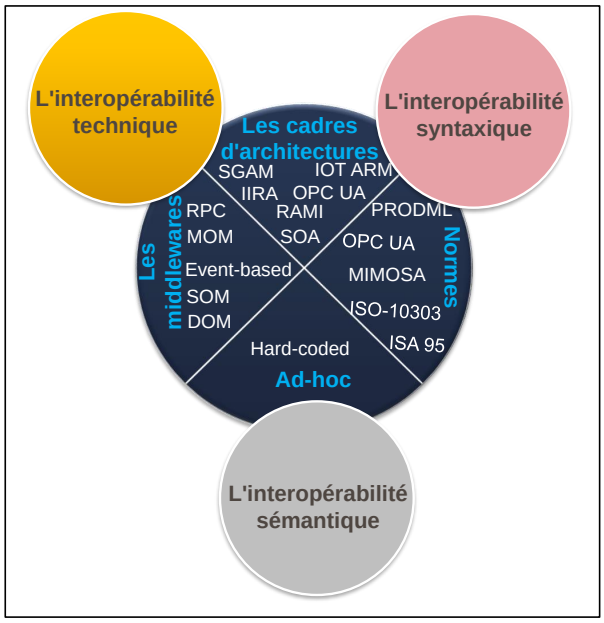


FIGURE 2.19 – Approches de l'interopérabilité multi niveaux

des normes et de la technologie (National Institute of Standards and Technology - NIST en anglais), qui introduit un cadre d'interopérabilité pour Smart Grid, a conclu qu'une centaine de normes doivent être interopérables pour assurer l'interopérabilité de bout en bout [Framework, 2010]. De même, dans les environnements IdO et l'industrie 4.0, plusieurs normes ouvertes seront nécessaires [Vermesan et al., 2011]. De plus, les normes doivent être retravaillées (et donc adaptées aux systèmes) chaque fois que de nouveaux problèmes techniques ou sémantiques doivent être pris en compte. Pour résumer, dans un tel environnement hétérogène, il n'est pas pratique d'imposer et de se conformer à un seul standard [Lewis et al., 2008, Razzaque et al., 2016b, Terziyan et al., 2010]. Toutefois, le cas échéant, le soutien des normes réduit considérablement la nécessité de développer des composants spécifiques.

Dans les architectures et les modèles de référence donnés par [CEN-CENELEC-ETSI and Coordination, 2012, Council, 2008, Hankel and Rexroth, 2015], l'interopérabilité de la communication et de l'information a été définie par des normes spécifiques au domaine. Dans un environnement hétérogène, la garantie préalable que tous les systèmes respectent le même accord de standard n'est pas accordé. L'architecture OPC UA [Mahnke et al., 2009] fournit une solution concrète pour gérer l'interopérabilité, cependant il faut toujours avoir des adaptateurs externes pour interopérer avec les systèmes non-OPC UA. De même, l'architecture SOA fournit une solution abstraite et concrète, mais il impose l'utilisation de technologie spécifique comme service web afin de gérer l'interopérabilité. L'architecture IOT-A, qui est présentée dans [Bassi et al., 2013], est très abstraite et fournit une approche pour modéliser un système IOT basé SOA. Cependant, il ne fournit pas une approche pour la mise en œuvre technique du système à l'étude. Les solutions comme [Varga et al., 2017] fournissent des so-

lutions techniques, cependant, il est applicable aux systèmes basés sur le service Web avec syntaxe et sémantique spécifiques. Cependant, dans notre contexte, il existe de nombreux systèmes non basés sur les services web (applications de bases de données, applications basées sur des fichiers, etc.) et différentes syntaxes et sémantiques doivent être gérées.

D'autres travaux s'appuient sur les middlewares. Les middlewares basés sur les appels de procédures à distance et basés sur les événements traitent de l'interopérabilité au niveau technique sans tenir compte de l'interopérabilité syntaxique et sémantique sauf quelques solutions comme OPC UA. Les middlewares orientés bases de données permettent la communication d'applications à différents types de bases de données. Les bases de données gèrent l'interopérabilité syntaxique et sémantique, mais l'interopérabilité technique est moins gérée. Par ailleurs, il y a plusieurs solutions qui sont basées sur les middlewares. Toutes les solutions imposent d'un commun accord d'utiliser par exemple, les techniques d'événements, les appels de procédure, les services, etc. Cependant, dans notre contexte et l'environnement hétérogène, nous ne pouvons pas imposer un commun accord. Il y a aussi plusieurs solutions/plate-formes académiques et commerciales pour gérer l'interopérabilité dans les environnements hétérogènes.

Dans la partie suivante, nous avons défini certains critères afin de comparer les différentes approches qui gèrent l'interopérabilité multi-niveaux.

- Générique : ce critère vérifie si la solution/l'approche est générique en termes d'application à un domaine particulier ou si elle est indépendante du domaine.

- Modulaire : ce critère vérifie si la solution offre une séparation des modules pour gérer chacun des trois niveaux d'interopérabilité séparément et indépendamment. La modularité simplifie le développement et la maintenance de chaque niveau d'interopérabilité séparément.

- Solution imposée : Pour chaque niveau d'interopérabilité, les techniques/mécanismes/normes imposés sont identifiés.

- Extensibilité : Pour chaque niveau d'interopérabilité, ce critère vérifie si les techniques/mécanismes/normes peuvent être étendus pour gérer d'autres techniques/mécanismes/normes.

- Règles métiers/opérationnelles soutien : ce critère vérifie si la solution/l'approche permet de gérer les règles métiers/opérationnelles.

- les portées d'interopérabilité : ce critère identifie la portée de la couverture d'interopérabilité pour l'échange de données : OT pour l'échange de données vertical ; IT pour l'échange de données horizontales.

			Technical interoperability		*Syntactic interoperability*		*Semantic interoperability*			
Feature / Solution	**Générique**	**Modularity**	**Imposed Soltuion**	**Extensibility**	**Imposed Solution**	**Extensibility**	**Imposed Solution**	**Extensibility**	**Buissnes/ operational rules**	**Interoperability scope**
IBM Watson IOT	Yes	No	MQTT, HTTP	No	JSON	No	Device Model	No	No support	OT
Azure IoT	Yes	No	AMQP, MQTT, HTTP, et API	No	JSON	No	No Support	N/A	N/A	OT
Oracle IOT cloud	Yes	No	HTTP/REST, HTTP/SOAP, WS-I et JMS	No	JSON	No	No Support	N/A	N/A	OT
SOCRADES	Yes	No	SOAP/Rest	Yes	Web service	No	1	1	1	OT
HYDRA	No	Yes	HTTP/REST, HTTP/SOAP	No	XML	No	Yes (Ontology) Integrative approach	UKnown	No support	OT
COOPERaTe	No	Yes	HTTP/REST, HTTP/SOAP	No	XML	No	Yes (NIM Model) Integrative approach	No	No support	OT
Xively		No	HTTP/REST, HTTP/SOAP	No	JSON, XML et CSV	No	No Support	N/A	N/A	OT
Arrowhead	No		HTTP/REST, HTTP/SOAP	Yes	XML	No	No Support	N/A	No	OT and IT
OpenIoT	No	No	Middleware	Yes	Uknown	-	Ontology (Integrative approach)	UKnown	No	OT
OPCUA	Yes	Yes	OPCUA	No	XML, Bytes	No	Agnostic Model	Yes	No	IT and OT
Middleware RPC	Yes	No	COM, DCOM, CORBA	No	No support	N/A	No Support	N/A	N/A	IT and OT
Event-Based Middlewares	Yes	No	Publish/subscribe protocls	No	No support	N/A	No Support	N/A	N/A	IT and OT
Middleware DOM	Yes	No	CLI	No	XML	No	Domain model Integrative approach	No	Yes	IT and OT

TABLEAU 2.1 – Évaluation des qualités de solution par rapport à l'état de l'art

Les solutions génériques sont indépendantes de domaines. Si la solution traite tous niveaux d'interopérabilités séparément et indépendamment, nous la considérons comme modulaire. Nous trouvons aujourd'hui que la plupart de solutions d'interopérabilité technique et syntaxique est orientée web service en utilisant les mécanismes HTTP/Rest, HTTP/SOAP, etc. avec des formats XMT et JSON. Quelques solutions gèrent l'interopérabilité sémantique via des modèles de données de domaine (Device ou NIM), les ontologies (comme Hydra). Cependant, il y a toujours la notion d'imposition de technologie ou normes. De plus, les solutions ne sont pas extensibles à tous les niveaux d'interopérabilité. Les solutions qui permettent de gérer les règles métiers/opérationnelles sont basées sur l'approche d'intégration qui n'est pas utile dans notre contexte. Enfin, la portée des solutions est soit l'échange de données au niveau OT ou au niveau IT. Ils ne peuvent pas être utilisés pour répondre aux besoins du marché de l'industrie 4.0 où tout doit être connecté à tout entre IT et OT. Pour résumer, les travaux antérieurs imposent une norme, un accord ou une technologie particulière pour promouvoir une solution d'interopérabilité. En fait, ces solutions sont très bonnes dans les contextes homogènes ou les systèmes se conforment à une norme, un accord ou une technologie unique. Ils peuvent être utiles dans un contexte hétérogène où le nombre de systèmes hétérogènes est limité, ce qui nécessite un nombre limité de composants ad-hoc. Nous pouvons supposer qu'ils sont rapides à se développer car ils suivent une approche d'intégration. Dans ce travail, nous cherchons une solution pour des contextes avec les systèmes complètement hétérogènes. Ainsi, les approches d'intégration ne sont pas utiles. Nous voulons mettre l'accent sur les caractéristiques d'une solution générique, modulaire à tous les niveaux de l'interopérabilité, agnostique qui n'impose pas une norme/accord/technologie spécifique pour les systèmes hétérogènes, extensibles qui peut être étendue à chaque niveau de l'interopérabilité, et qui permet de traiter les règles métiers/opérationnelles et qui favorise l'échange au niveau IT et OT.

2.3 L'ingénierie dirigée par les modèles

Au cours des dernières années, ce domaine de l'ingénierie logicielle a étudié comment différents formats de données et sémantiques peuvent être reliés et traités techniquement dans une architecture générique. Bien qu'il provienne de techniques de modélisation et de génération de logiciels, il a depuis été appliqué avec succès pour représenter et échanger des informations provenant du monde physique [Duddy and Steel, 2012].

L'ingénierie dirigée par les modèles (IDM), ou Model Driven Engineering (MDE) en anglais, est une approche de l'ingénierie logicielle qui favorise les « modèles » en tant que concept unificateur pour représenter l'information [Bézivin, 2005]. Le modèle est une représentation simplifiée d'une certaine réalité. Pour comprendre des problèmes complexes, nous devons les modéliser dans une représentation simplifiée, par conséquent, ils peuvent être facilement compris. Par exemple, un problème spécifique peut être modélisé en utilisant

des équations différentielles, des représentations graphiques, des représentations textuelles, etc.

Aujourd'hui, la technologie devient de plus en plus complexe. Par conséquent, les modèles deviennent un mécanisme puissant pour élever le niveau d'abstraction du développement logiciel. Le manifeste IBM définit trois idées essentielles à IDM [Bézivin, 2006, Booch et al., 2004] : représentation directe, automatisation et normes ouvertes. La première consiste à représenter les idées d'un domaine de problèmes en utilisant des modèles abstraits (langages spécifiques au domaine), en réduisant la distance sémantique entre le problème et les technologies pour les implémenter. Cela conduit à une productivité accrue et à des conceptions plus précises. La seconde utilise la transformation du modèle pour combler l'écart sémantique entre les différents concepts de domaine (spécification de haut niveau) et les technologies d'implémentation (code de niveau inférieur). En conséquence, cela augmente la vitesse et réduit les erreurs humaines pendant le processus. La dernière idée se concentre sur l'utilisation de normes ouvertes pour attirer les chercheurs, les fournisseurs d'outils et les utilisateurs industriels, permettant ainsi l'interopérabilité des solutions techniques.

De manière générale, IDM vise à augmenter la productivité, maximiser la compatibilité entre les systèmes, réduire la complexité, améliorer la qualité des logiciels, augmenter la formalisation, favoriser la séparation des préoccupations et la communication entre les développeurs et l'équipe travaillant sur un système.

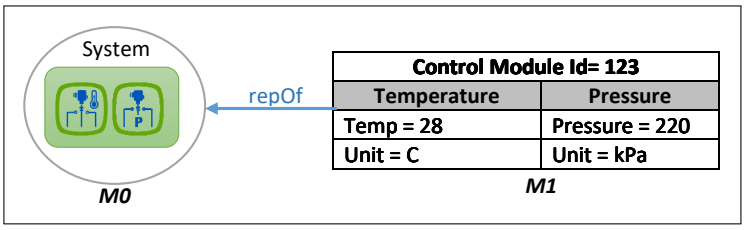


FIGURE 2.20 – La relation "repOf" entre le système (M0) et le modèle (M1)

La communauté distingue trois niveaux de modèles : modèle (terminal), métamodèle et métamétamodèle. Un modèle (M1) est une représentation d'un système pour véhiculer l'information d'une réalité (M0). La figure 2.20 illustre un exemple de modèle qui définit une représentation possible d'un module de contrôle qui mesure la température et la pression. La relation entre le modèle et le système qu'il représente est appelée "représentationDe" (repOf en anglais). Pour créer un modèle en bois, nous aurions besoin pour certains éléments tels que des arbres, des clous, des vis, etc. Ces éléments servent à créer les modèles et sont référencés par métamodèle. En IDM, un métamodèle est le langage de modélisation pour décrire un modèle. Par exemple, un module de contrôle est décrit par un identifiant et il est composé de capteurs de température et de pression comme montrés dans la figure 2.21. La relation entre le modèle et le langage utilisé pour le construire est appelée "conformeA" (conforms To en anglais). De même, un Métamodèle se conforme à un Métamétamodèle ; ce-

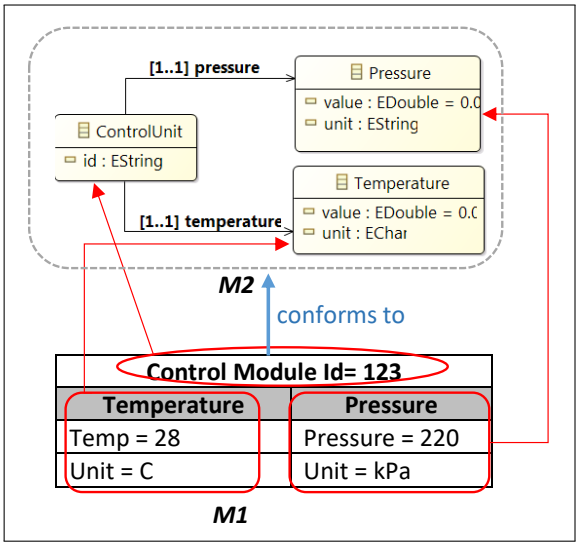


FIGURE 2.21 – La relation "conforms To" entre le modèle (M1) et son langage de modélisation (M2)

pendant, ce Métamétamodèle est également conforme à lui-même. L'OMG propose l'utilisation d'un métamétamodèle appelé MOF (Meta-Object Facility) qui relie les niveaux comme indiqué dans la figure 2.22. Il définit les concepts tels que la classe, les associations, les attributs, etc. Le métamodèle du module de contrôle est conforme au métametamodèle MOF comme indiqué sur la figure 2.23.

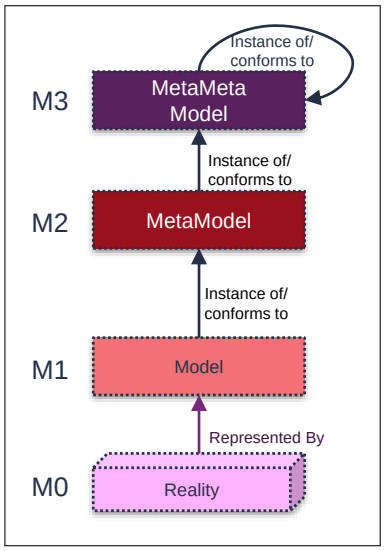


FIGURE 2.22 – La méta-modélisation à quatre couches

2.3.1 L'architecture dirigée par les modèles

L'architecture dirigée par les modèles, ou Model Driven Architecture (MDA) en anglais, est l'une des réalisations existantes pour les concepts et les principes MDE autour d'un

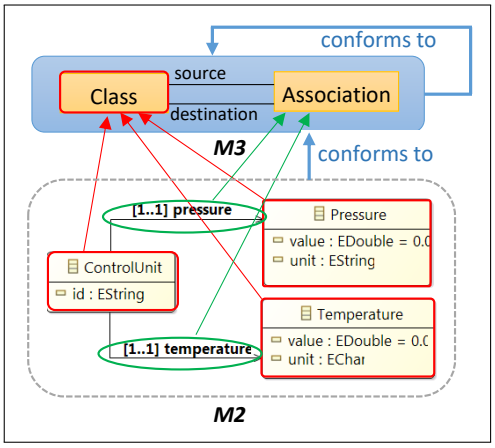


FIGURE 2.23 – La relation "conforms To" entre le métamodèle (M2) et son langage de modélisation (M3)

ensemble de standards OMG tels que Meta Object Facility (MOF) [OMG, 2013], XML Metadata Interchange (XMI) [OMG, 2007], Common Warehouse Meta-model (CWM), Unified Modelling Language (UML), Software Process Engineering Meta-model (SPEM), Query/View/Transformation (QVT), Object Constraint Language (OCL), etc. Il définit une approche pour séparer la spécification des fonctionnalités du système de la spécification de la mise en œuvre de ces fonctionnalités sur une plate-forme technologique particulière. Le MDA préconise l'élaboration de modèles selon trois niveaux d'abstraction tel qu'illustré dans la figure 2.24 : Computation Independent Model (CIM) pour représenter le domaine et les exigences d'un système dans lequel aucune considération informatique n'apparaît ; Platform Independent Model (PIM) pour modéliser la fonctionnalité du système (d'analyse et de conception) indépendamment des détails techniques des plate-formes d'exécution (J2EE, .Net, Web, etc.) ; Platform Specific Model (PSM) pour modéliser l'implémentation informatique par rapport à une plate-forme d'exécution particulière. L'un des objectifs majeurs du MDA est le passage de PIM à PSM de façon automatique via des mécanismes de transformation de modèles afin d'obtenir un gain significatif de productivité.

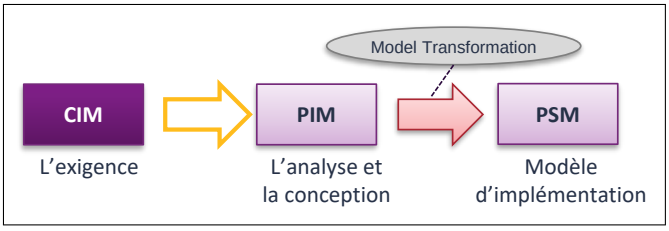


FIGURE 2.24 – Le processus de base MDA

2.3.2 La transformation de modèles

La transformation (cf. figure 2.25) de modèles est le processus de production d'un ou plusieurs modèles de sortie (Model B) à partir de l'un ou de plusieurs modèles d'entrée (Model A) tout en conservant la conformité avec le métamodèle (MetaModel A, MetaModel B). Si les métamodèles (MetaModel A et MetaModel B) sont différents, la transformation est de type exogène sinon la transformation est de type endogène [Combemale, 2008].

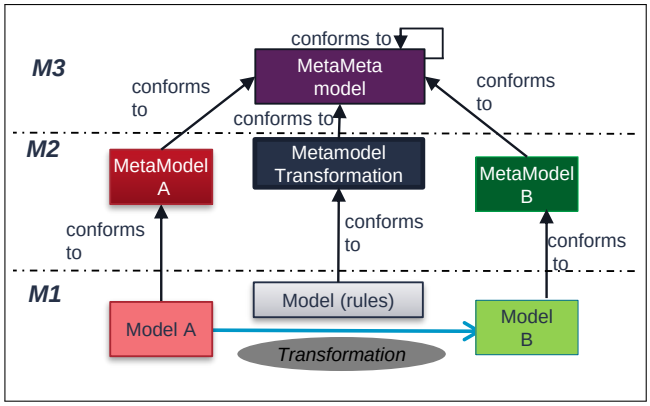


FIGURE 2.25 – le processus de transformation du modèle

La transformation de modèles joue un rôle important pour [Czarnecki and Helsen, 2006] : (i) générer des modèles de niveau inférieur à partir de modèles de niveau supérieur et vice versa (transformation verticale) ou générer des modèles dans le même niveau (transformation horizontale). Dans le MDA, la transformation verticale peut être illustré dans le cas du passage de PIM à PSM tandis que la transformation horizontale dans le cas de transformation PIM à PIM ou PSM à PSM (cf. figure 2.26) [Gerber et al., 2002] ; (ii) créer de différents points de vue selon les différentes préoccupations du système ; (iii) ré-usiner les modèles (model refactoring en anglais). Le réusinage est le processus consistant à améliorer la structure d'un artefact logiciel sans modifier son comportement prévu [Fowler et al., 1999]. Par exemple changer le modificateur d'un attribut de "public" à "private" ; (iv) interopérer les systèmes différents.

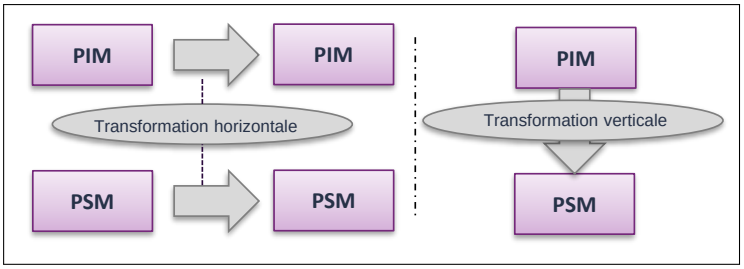


FIGURE 2.26 – Les types de transformations [Combemale, 2008]

Pour mieux comprendre la transformation de modèle, l'exemple de module de contrôle (cf. 2.21) sera transformé en un autre modèle dédié uniquement à la température en Fahrenheit (transformation horizontale). Le but est de gérer un modèle (point d'interrogation) conforme au metamodèle "TempModule" comme le montre la Figure 2.27.

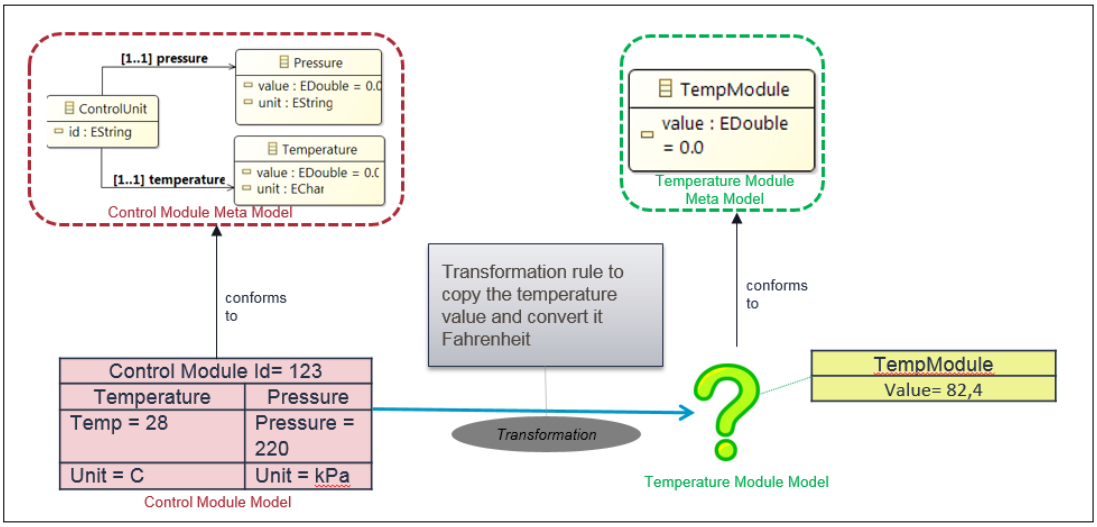


FIGURE 2.27 – Exemple de transformation de modèle

Les transformations de modèles peuvent être réalisées en utilisant un langage de programmation pour des manipulations directes de modèles. Cependant, l'approche la plus appropriée consiste à utiliser un langage spécifique au domaine (Domain Specific Language - DSL en anglais) conçu spécialement pour la manipulation de modèles tels que : ATL [Jouault et al., 2008, Jouault and Kurtev, 2006], QVT [OMG, 2011], ETL [Kolovos et al., 2008], etc. Un moteur de transformation exécute les règles de transformation spécifiées par le développeur pour obtenir le modèle de sortie souhaité. Dans certains problèmes complexes, une transformation n'est pas suffisante et il est nécessaire d'exécuter une chaîne de transformation de modèle [von Pilgrim et al., 2008] qui permet le développement, le test, et la réutilisation de la transformation individuelle du modèle.

2.3.3 Espace technique

Un espace technologique (ET), Technical Space (TS) en anglais, est un contexte de travail avec un ensemble de concepts associés, de connaissances, d'outils, de compétences requises et de possibilités. Il est souvent associé à une communauté d'utilisateurs avec un savoir-faire partagé, un soutien éducatif, une littérature commune et même des réunions d'ateliers et de conférences. C'est à la fois une zone d'expertise établie et de recherche en cours et un référentiel de ressources abstraites et concrètes. La figure 2.28 illustre quelques exemples d'espace technique tels que [Kurtev et al., 2002] : (i) L'espace technique de la syntaxe abstraite du langage de programmation, où un programme est écrit dans un langage de programmation

dont la syntaxe est spécifiée dans une grammaire; (ii) L'espace technique de XML, où XML est une norme pour la représentation des données. Les documents XML sont écrits en syntaxe et validés en fonction des contraintes de bonne forme et de validité données dans leur langage de schéma (XML Schema language, Document Type Definition - DTD); (iii) l'espace technique de systèmes de gestion de bases de données (DBMS), où les données de la table de base de données sont conformes à leur schéma de base de données (structure) (iv) l'espace technique de MDA (cf. 2.3.1).

Les techniques de transformation peuvent être appliquées de manière interne dans un espace technique, mais elles peuvent également être appliquées à travers des limites d'espace techniques (les ponts entre des ES). Dans le dernier cas, cette transformation est appelée "Projection" afin de les distinguer des transformations internes à un espace technique. Cette projections permet de relier différents TS afin de gérer les problèmes d'interopérabilité entre eux. Avec la projection, la conception commence dans un ET et le résultat souhaité est obtenu par transformation pour un autre ET. Par exemple, un document XML dans le XML TS est projeté dans MDA TS. En relation avec le TS MDA, la projection est divisée en deux types types d'opérateurs : injecteur et extracteur. L'injecteur est la transformation des données de non MDA-ET (XML, base de données, fichiers plats) en MDA (modèles) tandis que l'extracteur est l'inverse. Celles-ci peuvent être implémentées manuellement ou facilitées en utilisant des outils XML ou grammaticales [Efftinge and Völter, 2006].

En résumé, lorsque la transformation est effectuée de manière interne dans un espace technique, elle est appelée "Transformation", cependant, lorsque la transformation est effectuée à travers les limites de l'espace technique, elle est appelée "Projection".

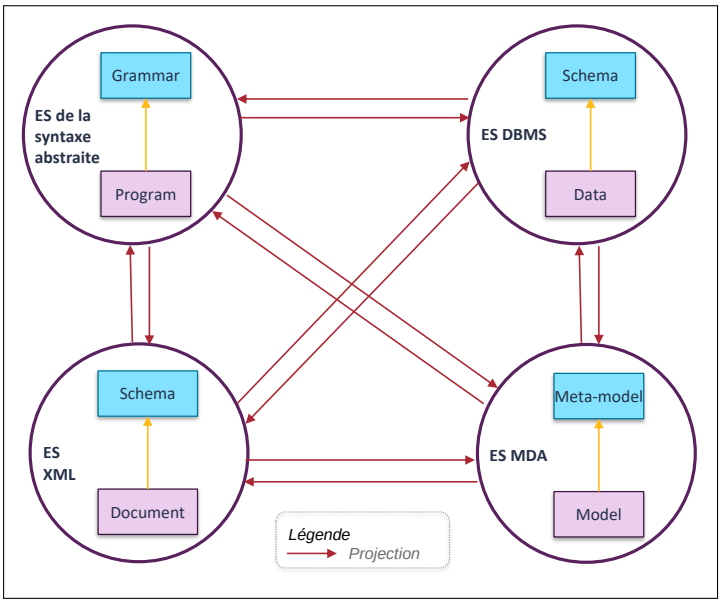


FIGURE 2.28 – Des ponts entre l'espace technique [Kurtev et al., 2002]

2.3.4 Approches existantes d'interopérabilité par l'IdM

Certaines initiatives et approches gèrent l'interopérabilité à l'aide de IDM tels que les projets **ATHENA IP** (Advanced Technologies for interoperability of Heterogeneous Enterprise Networks and their Applications - Integrated Project) et **INTEROP NoE** (Interoperability Research for Networked Enterprises Applications and Software - Network of Excellence) [24]. Ces projets fournissent des cadres d'interopérabilité qui favorisent l'utilisation des modèles pour le développement de systèmes et d'applications. Ces types de cadres, qui utilisent des modèles pour gérer l'interopérabilité, sont référés par "l'interopérabilité dirigée par modèle (Model Driven Ineroperabiltiy - MDI an anglais)" [Bourey et al., 2007].

Athena IP utilise le cadre d'interopérabilité INTEROP NoE dans certaines parties de son cadre. Nous présenterons une description d'introduction du projet, plus de détails sont disponibles dans [Berre et al., 2007, Elvesæter et al., 2006]. Le cadre "ATHENA Interoperability Framework (AIF)" [Berre et al., 2007] est le résultat synthétisé du projet. Il est structuré en trois parties :

- **L'intégration conceptuelle** axée sur les concepts, les métamodèles, les langages et les relations entre les modèles. Il fournit une base pour la systématisation de divers aspects de l'interopérabilité des modèles logiciels.
- **L'intégration technique** axée sur les environnements de développement et d'exécution de logiciels. Il nous fournit des outils de développement pour développer des modèles de logiciels et des plates-formes d'exécution pour exécuter des modèles de logiciels.
- **L'intégration applicative** axée sur les méthodologies, les normes et les modèles de domaine. Il nous fournit des directives, des principes et des modèles pouvant être utilisés pour résoudre les problèmes d'interopérabilité des logiciels.

Pour chaque partie, il existe une référence pour décrire et soutenir l'application de l'IDM. Ce cadre est utile pour gérer l'interopérabilité entre les entreprises. Il prend en compte la vision d'entreprise supérieure du processus et du contexte de l'entreprise, jusqu'à la vue inférieure du code technique des systèmes et des applications. Il fournit des directives sur l'utilisation des normes et des ontologies aux différents niveaux. Autrement dit, ce cadre, qui est axé sur les modèles, met l'accent sur l'interopérabilité des applications/systèmes de l'entreprise via avec une approche d'intégration et l'utilisation des normes et des ontologies. Il peut être utilisé dans un contexte où l'environnement ne change pas beaucoup. Cependant, dans un contexte d'hétérogénéité qui ne peut pas se conformer à la norme unique et dans un environnement qui évolue fréquemment, l'approche d'intégration n'est pas compatible, car l'interopérabilité des systèmes doit être modifiée à chaque changement d'environnement.

[24] http ://www.interopnoe.org/

D'autres approches utilisent aussi l'IDM pour assurer l'interopérabilité. [Guillén et al., 2013] gére l'interopérabilité dans le contexte de la technologie de Cloud Computing. Selon [Mell and Grance, 2011], le Cloud Computing est un modèle qui permet un accès réseau et sur demande à un pool partagé de ressources informatiques configurables (par exemple, des réseaux, des serveurs, du stockage, des applications et des services) qui peut être rapidement approvisionné et disponible sans trop d'efforts de gestion ou d'interaction d'opérateurs. Ce modèle de cloud favorise la disponibilité et est composé de cinq caractéristiques essentielles, de trois modèles de service et de quatre modèles de déploiement. Les fournisseurs de service de cloud sont des fournisseurs qui louent à leurs clients des services de Cloud Computing pour leurs applications cloud. Cependant, ces fournisseurs définissent des services propriétaires sans tenir compte de leur interopérabilité avec ceux définis par leurs concurrents. Ainsi, les applications cloud des clients sont liées aux services des fournisseurs propriétaires. [Guillén et al., 2013] propose un cadre "MULTICLAPP" qui utilise l'IDM en favorisant la séparation des préoccupations pour interopérer l'application cloud sur les systèmes des différents fournisseurs de cloud. Autrement dit, une application cloud peut être développée indépendamment du fournisseur de services cloud. Dans ce travail, nous ne gérons pas l'interopérabilité entre les applications cloud et les systèmes de fournisseurs de cloud, mais plutôt entre les systèmes hétérogènes. [Bondé et al., 2006] utilise l'IDM pour assurer l'interopérabilité entre les modèles dans l'espace technique des models. [Del Fabro et al., 2006] utilise le IDM pour gérer l'interopérabilité sémantique pour l'intégration de données.

2.3.5 Synthèse

L'IDM fournit de nombreuses techniques pour les opérations de modèles qui peuvent être utilisées pour réaliser l'interopérabilité dans notre contexte tel qu'illustré dans la figure 2.29. Au niveau syntaxique, *projection* permet de convertir des formats de données de/vers des espaces techniques spécifiques (ET) vers/depuis l'environnement de modélisation. Celles-ci peuvent être implémentées manuellement ou facilitées en utilisant des outils XML ou grammaticaux [Efftinge and Völter, 2006]. Au niveau sémantique, *transformations* [Czarnecki and Helsen, 2006] permettent de mapper des données d'une sémantique d'un métamodèle à un autre. De tels métamodèles peuvent être spécifiques à un domaine, basés sur des normes, ou fournir une sémantique unifiée personnalisée. La figure montre une exemple de ces opérations.

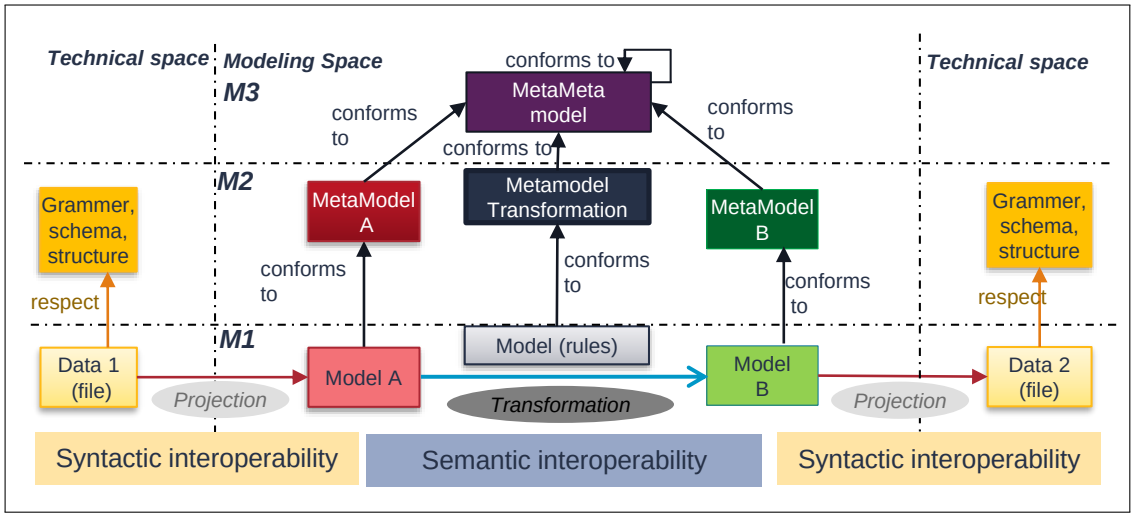


FIGURE 2.29 – Les opérations de projections et transformations

2.4 Conclusion

Cet état de l'art inclut plusieurs solutions pour soutenir le problème d'interopérabilité afin d'agréger et échanger les données entre des systèmes hétérogènes. Ce problème, dont l'ampleur ne cesse pas de croître avec le temps, est souvent abordé en imposant une norme ou une technologie à respecter par tous les systèmes. Cependant, cela ne peut pas être toujours réalisé dans ce monde de systèmes hétérogènes.

Nous avons présenté des solutions abordant chaque niveau d'interopérabilité séparément. Cependant, elles ne sont pas suffisantes dans notre contexte où les trois niveaux doivent être abordés. Ensuite, nous avons présenté les solutions qui prennent en compte plusieurs niveaux de l'interopérabilité comme les architectures, les normes et les middlewares. Nous avons comparé les solutions sur certains critères, nous avons conclu que toutes les solutions imposent des normes/techniques à respecter afin de promouvoir l'interopérabilité. Nous avons montré le degré de modularité, l'extensibilité et la portée de chaque solution. Quelques solutions sont modulaires pour traiter chaque niveau de l'interopérabilité, mais ils ne sont pas extensibles pour gérer d'autres normes/techniques. Ces solutions sont utiles dans des contextes homogènes, mais nous abordons des contextes de systèmes complètement hétérogènes dans ce travail. Nous avons clôturé le chapitre en représentant les concepts de IDM et comment il peut être utilisé pour gérer l'interopérabilité.

Le chapitre suivant illustre notre proposition afin de gérer l'interopérabilité en temps réel entre les systèmes hétérogènes de manière générique, modulaire, extensible et agnostique. Nous nous appuierons sur ces dernières caractéristiques pour valider notre proposition dans chapitre 6.

Deuxième partie

Proposition

Chapitre 3

Architecture Smart-hub

Sommaire

Développer une solution pour un système complexe, tel qu'un logiciel d'interopérabilité, est une tâche difficile. Alors, comment allons-nous concevoir une solution pour notre problème défini au chapitre 1? Selon [Budgen, 2003], après la définition des exigences, le processus de conception implique : la conception d'un modèle de boîte noire du problème; la proposition d'un modèle de boîte blanche; la mise en œuvre et l'évaluation de la solution par rapport aux exigences.

Dans la première section 3.1, nous commençons par un exemple qui nous servira de fil rouge pour suivre facilement la proposition et les chapitres suivants. La deuxième section présentée en section 3.2 est une représentation de la solution comme une boîte noire en relation avec son environnement. C'est une analyse fonctionnelle externe de la solution. Elle est utilisée pour décrire ce qu'un élément va faire, plutôt que comment il va le faire.

La troisième section présentée en section 3.3 est une réalisation détaillée de la solution. C'est une analyse fonctionnelle interne détaillée de la solution. Autrement dit, elle est utilisée pour décrire comment le système va le faire.

3.1 Exemple d'étude

Cet exemple sera utilisé dans tout le document pour illustrer la proposition sur un scénario simplifié d'échange de données dans un environnement de supervision de réseaux de gaz. Il est composé des systèmes suivants, comme indiqué dans la figure 3.1.

- **SCADA1** : il surveille la qualité et la quantité de gaz à certains points du réseau de gaz en temps réel. Il utilise le protocole OPC UA pour communiquer et représenter les données. Il est basé sur l'option d'interaction "Pull Aperiodic Unicast".
- **SCADA2** : il fournit des informations d'intensité lumineuse en temps réel. Les informations sont disponibles via un service web (Rest) à partir d'un serveur back-end Sigfox Cloud. La syntaxe est décodée en 12 Byte et la sémantique est propre au système . Il est basé sur l'option d'interaction "Pull Periodic Unicast".
- **GMAO** : il fournit les informations de maintenance et de géolocalisation pour les analyseurs et les capteurs de lumière. Les informations sont disponibles sous la forme d'un fichier tabulaire partagé (Excel format). Il est basé sur l'option d'interaction "Pull Periodic Unicast".
- **SIG** : il affiche les informations de SCADA1 (par exemple CH4) et SCADA2 (par exemple intensité de lumière) sur leur emplacement approprié sur la carte en temps réel. En outre, les règles opérationnelles sont nécessaires sur les données des capteurs avant de pouvoir être présenté au SIG. Les données doivent être fournies par un fichier tabulaire partagé (format CSV). Il est basé sur l'option d'interaction "Push Periodic Unicast".

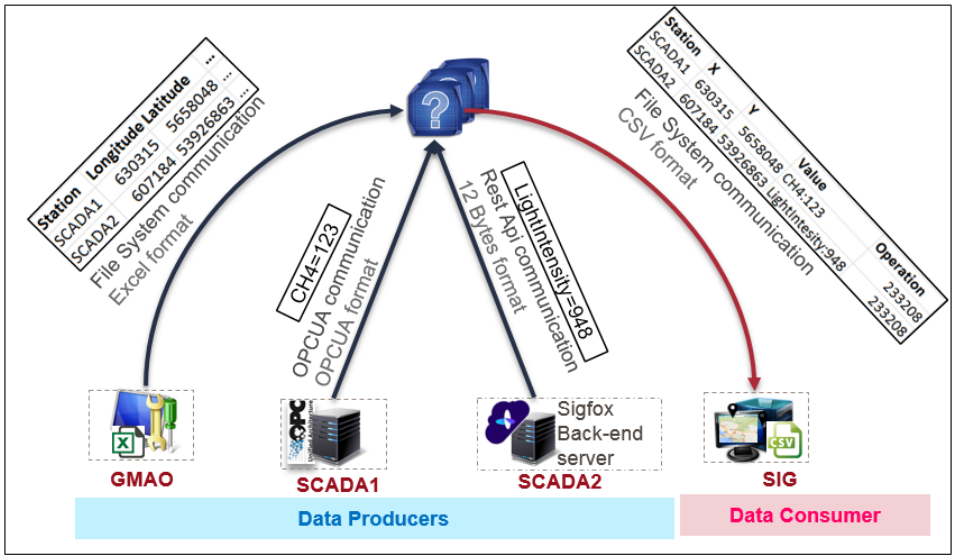


FIGURE 3.1 – Des systèmes dans une exemple d'étude

3.2 Boîte noire : analyse fonctionnelle externe du Smart-hub

La mission de notre proposition est de promouvoir l'interopérabilité entre les systèmes hétérogènes sans imposer une norme. La solution vise à agréger les données de systèmes (producteur de données), les traiter et les rendre disponibles pour d'autres systèmes externes consommateurs de données. Donc, les principaux acteurs de l'environnement extérieur qui interagissent avec notre proposition sont (cf. figure 3.2) :

- **Les producteurs de données**, Data producers en anglais, sont des systèmes externes qui mettent en disposition ses données ou fournissent un moyen d'accéder aux données pour être utilisées par un autre système. Par exemple, les données de systèmes GMAO, SCADA1, et SCADA2 sont mises à disposition via un fichier, interface OPC UA et webservice, respectivement .
- **Les consommateurs de données**, Data consumers en anglais, sont des systèmes externes qui utilisent les données qui lui sont fournies via des différents moyens et interfaces. Par exemple, les données de systèmes SIG sont fournies via un serveur de fichier qui héberge un fichier CSV.

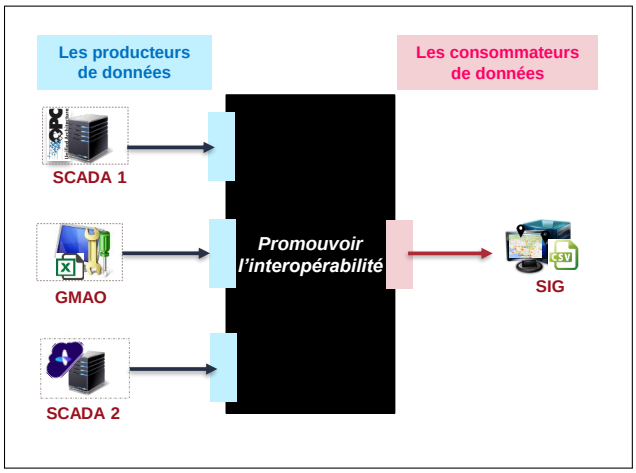


FIGURE 3.2 – Diagramme de contexte décrivant les interactions entre la proposition et l'environnement extérieur.

3.3 Boîte blanche : analyse fonctionnelle interne du Smart-hub

La réalisation détaillée de notre solution est présentée dans cette section. En effet, lors du développement d'un système complexe, il est fondamental de spécifier correctement et clairement son architecture logicielle [Taylor et al., 2009].

Avant de commencer notre proposition d'architecture, il faut poser cette question : qu'est-ce qu'une architecture logicielle? Il y a plusieurs définitions pour répondre à cette

question. [Team, 2009] le définit comme le processus de définition d'une solution structurée qui répond à toutes les exigences techniques et opérationnelles, tout en optimisant les attributs de qualité communs tels que les performances, la sécurité et la gestion. IEEE [IEE, 2007] définit l'architecture comme l'organisation fondamentale d'un système, incarnée dans ses composants, leurs relations les uns aux autres et avec l'environnement, ainsi que les principes régissant sa conception et son évolution. Une autre définition donnée par [Bass et al., 2012] est la structure ou les structures du système, qui comprennent des éléments logiciels, les propriétés visibles à l'extérieur de ces éléments et les relations entre eux. Toutes ces définitions nous amènent à comprendre simplement l'architecture comme la décomposition d'un système en parties [Fowler, 2002], qui sont liées pour accomplir les fonctions définies.

Les paradigmes de développement, tels que Component Based Software Development -CBSD [Pree, 1997] (également appelé Component-based software engineering - CBSE [Brown and Wallnau, 1998, Crnkovic, 2001]), facilitent la création de systèmes évolutifs, flexibles, fiables et réutilisables. CBSE a des implications notables sur l'architecture d'un système. Une architecture qui prend en charge CBSE, par exemple en imposant l'utilisation d'un modèle de composant, est appelée une architecture logicielle basée sur les composants. Il met l'accent sur la séparation des préoccupations par rapport à la large gamme de fonctionnalités disponibles dans un système logiciel donné. De plus, il repose sur le concept de construction des composants qui sont destinés à être indépendants et réutilisables. Ces composants sont des unités logicielles indépendantes du contexte, aussi bien dans le domaine conceptuel que dans le domaine technique. Chaque composant fournit un certain type de service dans le contexte de l'application.

Comment ces composants seront extensibles? Un cas particulier de CBSD est l'application basée sur le plug-in [Mayer et al., 2003]. Ces applications permettent à des développeurs tiers de créer ou d'étendre ses capacités à l'exécution par des modules chargés dynamiquement ou des classes inconnues lors de la compilation. Les extensions aux applications sont appelées plug-ins [Szyperski, 2002]. L'un des avantages de l'application basée sur les plug-ins est l'amélioration de la conception. Cela permet aux développeurs et aux tiers d'étendre les capacités du composant principal pour ajouter/mettre à jour de nouvelles fonctionnalités via plug-ins sans toucher au composant de base. En fait, le code du composant de base est séparé du code du composant d'extension (le plug-in). Ainsi le code de ces composants et plug-ins est plus flexible, modulaire et maintenable.

Le composant de base de l'application doit être conçu pour accepter les plug-ins et fournir les spécifications de conception qui permettent aux développeurs d'écrire des plug-ins sans avoir directement accès au code source ou aux mécanismes internes de l'application primaire. L'application primaire devrait permettre l'utilisation de nombreux plug-ins en parallèle et permettre une extension du système en composant plusieurs plug-ins ensemble. Les plug-ins sont largement utilisés dans des applications telles que Photoshop par Adobe, où des effets d'images supplémentaires, des filtres et des tâches plus performantes sont ajou-

tés pour étendre la compatibilité de l'application. Les plug-ins sont également utilisés par les navigateurs web pour ajouter de nouvelles fonctionnalités, par exemple, ajouter le support de flash player ou shockwave. Netscape Communicator, des systèmes de gestion de contenu courants tels que WordPress ou Drupal, etc. sont des exemples d'applications utilisant des plug-ins.

Dans la partie suivante, nous proposons une architecture logicielle modulaire et extensible en répondant à nos objectifs. Il permet de gérer le problème d'interopérabilité entre les systèmes dans un environnement moderne à grande échelle.

3.3.1 L'architecture "Smart-hub"

L'architecture proposée appelée "Smart-hub" (qui répond à notre objectif F5) est composée de six composants, qui sont organisés en quatre composants horizontaux (composant de communication, composant spécifique au domaine, composant indépendant du domaine et composant du système de systèmes) et deux composants verticaux (composant d'orchestration et composant de paramétrage) tel que montré par la figure 3.3. L'architecture agrège les données de plusieurs systèmes (producteurs de données) et les rend disponibles pour des systèmes externes consommateurs de données. Le but est d'interagir avec chaque système de la façon dont il est fourni et sans imposer une norme ou une technologie spécifique.

3.3.1.1 Composant de communication

Dans un environnement industriel, les systèmes hétérogènes (tels que SCADA1, SIG, GMAO, SCADA2 etc.) utilisent dés différents mécanismes de communication et interface (OPC UA, FileSharing, Rest Webservices, etc.). Certains systèmes ne sont pas activés pour la communication, par conséquent, c'est le rôle de ces systèmes de permettre la communication via des passerelles pour des exemples. Afin d'interagir avec un système ou la passerelle du système, la première étape est d'établir la communication avec le système selon sa méthode de communication. Donc, ce composant, Communication component en anglais, gère plusieurs et différents mécanismes de communication (sous-composants) afin d'établir la communication et d'échanger des données avec ces systèmes. D'une part, il lit les données de DP et les met à disposition pour le Smart-hub. D'autre part, il met à disposition des données du Smart-hub pour le consommateur de données.

Ces mécanismes ne sont pas fixés ou prédéfinis, donc, ils peuvent changer selon l'environnement. De plus, nous ne pouvons pas restreindre ce composant à certains mécanismes de communication. Par conséquent, nous proposons d'ajouter la notion et l'approche de plug-ins au composant de communication. Ce composant déclare un point d'extension où d'autres composants peuvent se brancher. Cela permet de combiner plusieurs mécanismes et répondre à notre objectif F1. Reprenons notre exemple fil rouge, nous ajoutons les points

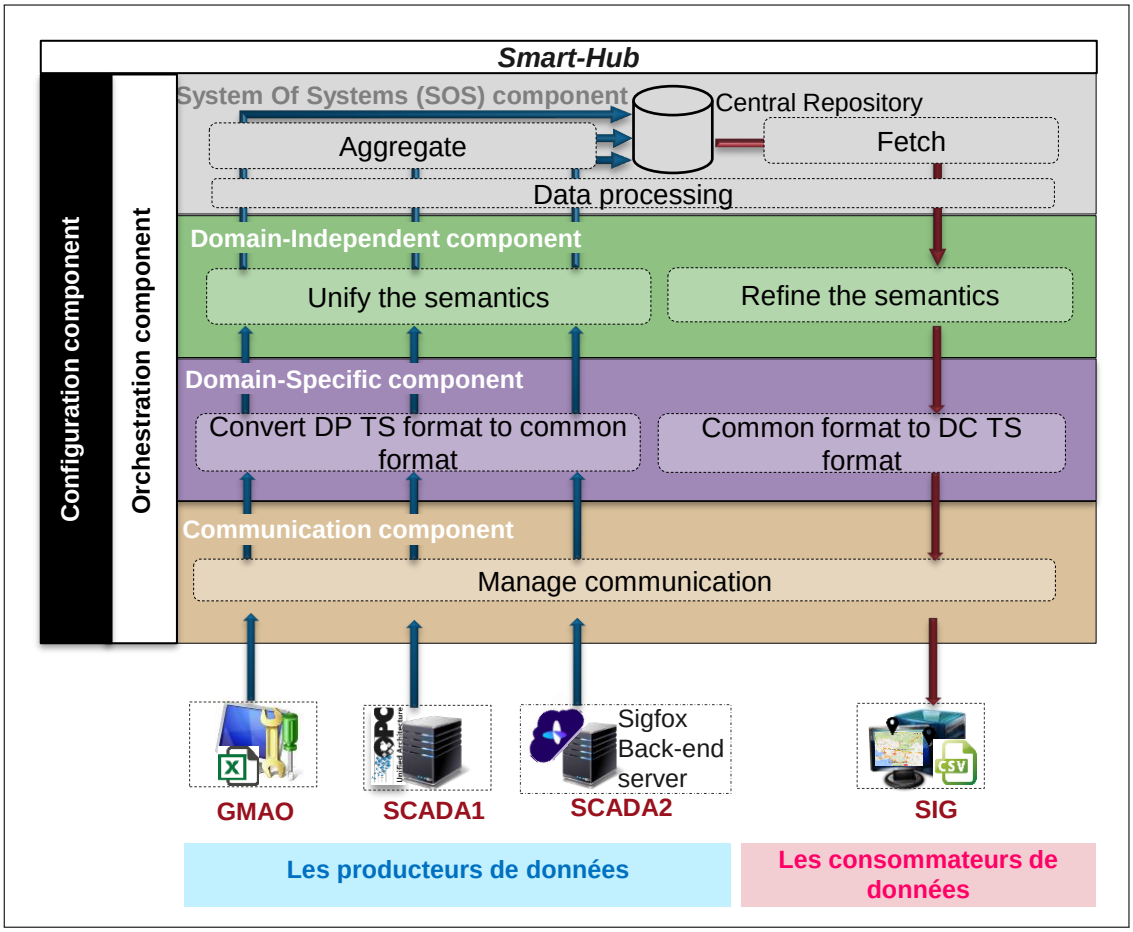


FIGURE 3.3 – Architecture conceptuelle : Smart-hub

d'extension pour OPC UA, web service, et le connecteur de fichier afin de communique avec les systèmes SCADA1, SCADA2 et GMAO/SIG.

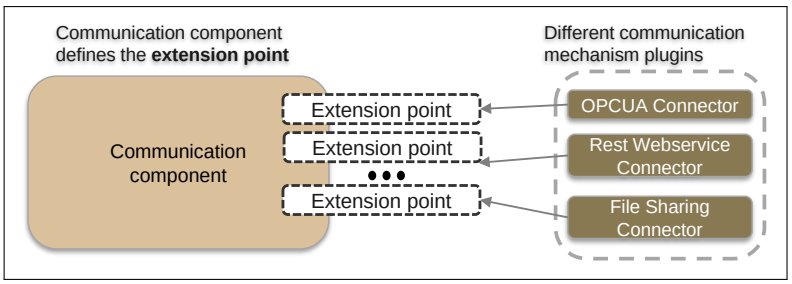


FIGURE 3.4 – Composant de communication

3.3.1.2 Composant spécifique au domaine

Les systèmes utilisent différents formats pour la représentation de données tels que XML, XMI, JSON, texte, octets, etc. Ce composant, Domain-Specific component en anglais, permet

de gérer différents formats en répondant à notre objectif F2. Il convertit les données fournies par les producteurs de données du format dans un espace technique en un format commun dans un autre espace technique. De même, il convertit les données du format commun au format dans l'espace technique du consommateur de données. Ce processus de conversion est réalisé via des règles de conversion. Ces processus de conversion ne touchent pas les sémantiques de données et ils laissent la sémantique spécifique au domaine/système. Pour résumer, ce composant garde les règles de conversion et les sémantiques de systèmes.

Encore ici, nous ne pouvons pas restreindre le composant à des règles de conversion prédéfinies. Donc, nous proposons d'ajouter la notion et l'approche de plug-in au composant spécifique au domaine afin de permettre aux développeurs d'ajouter leurs règles de conversion (comme extensions) en fonction de l'environnement existant. La figure 3.5 illustre les listes de plug-ins pour gérer les formats de systèmes dans notre exemple de fil rouge : Excel_CommonFormat pour convertir les données de format Excel vers un format commun; OPUCA_Commonformat pour convertir les données de format XML de OPC UA vers un format commun, etc.

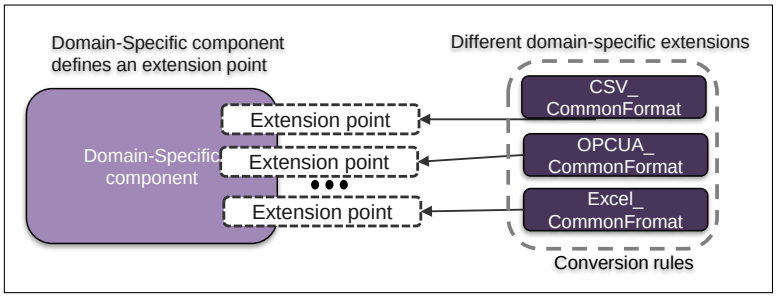


FIGURE 3.5 – Composant spécifique au domaine

3.3.1.3 Composant indépendant du domaine

Dans un environnement industriel, les systèmes interprètent les données différemment à cause des différentes sémantiques (OPC UA, CIM, système propre, etc.). Donc nous proposons d'ajouter ce composant - Domain-Independent component en anglais. Son but est de traiter et d'interopérer les différentes sémantiques des données. Nous proposons l'utilisation de l'approche d'unification pour gérer l'interopérabilité sémantique. Dans cette approche, une "sémantique commune" est choisie comme modèle pivot et les règles de mapping gèrent l'association sémantique entre ce modèle pivot et le modèle de système. Dans la figure 3.6, nous définissons deux types de règles de mapping : (i) les règles pour unifier les sémantiques (unify), c'est-à-dire les règles pour lier la sémantique de DP avec le modèle pivot; (ii) les règles pour raffiner les sémantiques (refine), c'est-à-dire les règles pour lier le modèle pivot avec la sémantique de DC. Pour résumer, ce composant doit définir un modèle pivot et

gère les différentes règles d'associations sémantiques entre ce modèle et les sémantiques de systèmes (DP et DC).

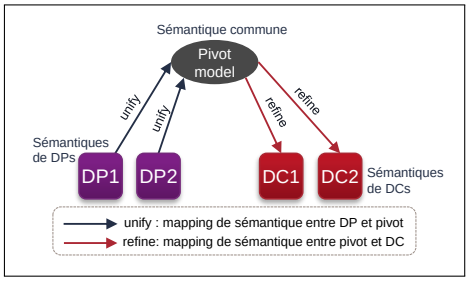


FIGURE 3.6 – L'approche unification dans le SH

Étant donné que ces règles de mapping ne sont pas prédéfinies et nous ne voulons pas restreindre le composant à des règles spécifiques, nous ajoutons aussi la notion et l'approche de plug-in dans ce composant. Par exemple (cf. figure 3.7), nous ajoutons les plug-ins pour faire le mapping entre les données OPC UA en format commun et les données en sémantique commune. Dans la même façon, nous ajoutons le plug-in pour faire le mapping de données entre un format commun "tableau" vers les données en sémantique commun, etc.

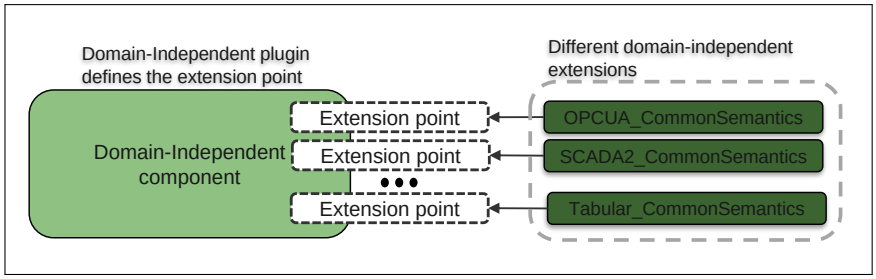


FIGURE 3.7 – Composant indépendant du domaine

3.3.1.4 Composant du système de systèmes

Ce composant, System of Systems component en anglais, est le lieu de dépôt, de retrait, et de traitement de données. Cela implique trois opérations comme illustrées dans la figure 3.8 : agréger (Aggregate), retirer (Fetch), traiter les données (Data Processing).

Agréger (Aggregate) :

Cette opération permet d'agréger les données qui sont unifiées dans le composant indépendant du domaine dans un référentiel central. Le référentiel central est le point de dépôt pour les données de tous les DP.

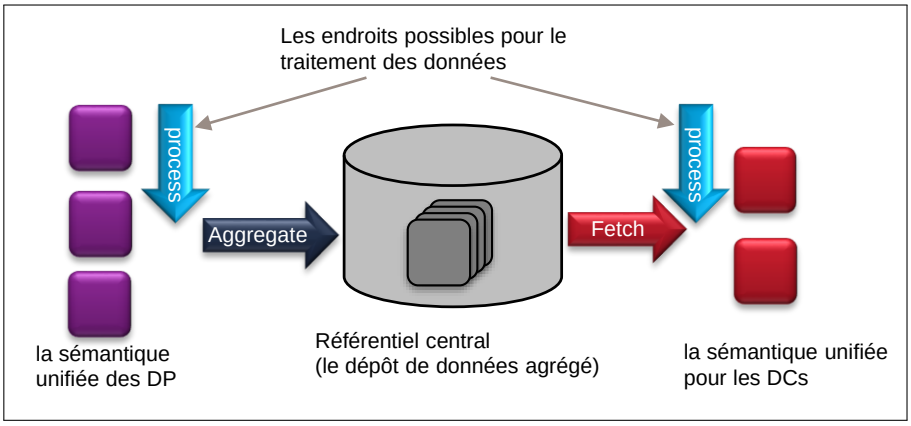


FIGURE 3.8 – Les opérations dans le composant SOS

Retirer (Fetch) :

Le référentiel central est la principale source de données pour les consommateurs de données. Ainsi, cette opération permet de chercher et de retirer des données de ce référentiel central. Ensuite, les données retirées sont mises à disposition pour un traitement ultérieur ou pour le composant indépendant du domaine.

Traiter les données (Data Processing) :

Cette opération permet de traiter les données en appliquant des règles métiers/opérationnelles telles que calculer la moyenne ou la somme sur les données de plusieurs DPs, renommer une variable, etc. En fait, cette opération est une étape complémentaire pour gérer les sémantiques pour les DC. Un système DP utilise l'unité de température Celsius, mais le DC utilise l'unité de température Fahrenheit. Les opérations de traitement de données peuvent être appliquées dans deux endroits : (i) avant l'agrégation (pre-agrégation) ou (ii) après l'agrégation (post-agrégation).

Cela peut être clarifié par un exemple, le DC (SIG) doit utilise une valeur Z qui est le double de la somme de données X et Y (Z= (X+Y)*2) qui sont fournies par les deux DPs SCADA1 et GMAO respectivement. Les figures 3.9 et 3.10 illustrent les deux approches pre-agrégation et post-agrégation pour effectuer cette opération (Z= (X+Y)*2).

- **l'approche pre-agrégation** : après les données (X, Y) sont disponibles, elles sont traitées et le résultat (Z) est agrégé dans le référentiel central. Ensuite, l'opération (Fetch) retire et met à disposition les données déjà traitées (Z=6) pour le DC (cf. la figure 3.9) .

- **l'approche post-agrégation** : après les données (X, Y) sont disponibles, elles sont agrégées dans le référentiel central. Ensuite, les données sont retirées (fetch) pour effectuer l'opération (Z= X+Y*2) et sont mises à disposition pour le DC (cf. la figure 3.10).

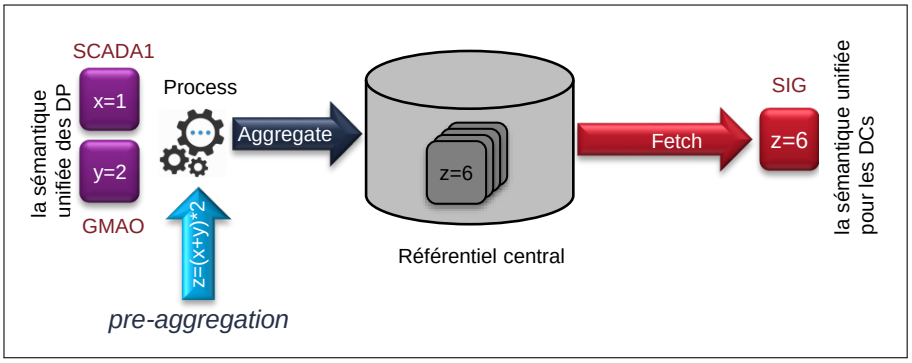


FIGURE 3.9 – Traitement des données : pre-agrégation.png

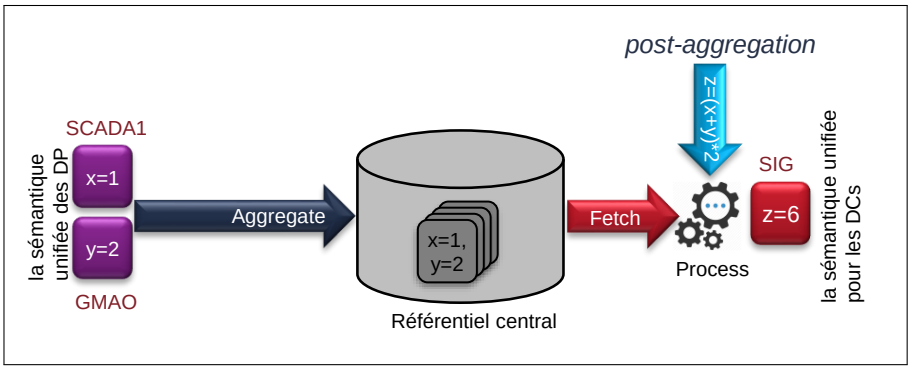


FIGURE 3.10 – Traitement des données : post-agrégation.png

Les règles métiers/opérationnelles changent selon le cas de usage et l'environnement dans lesquelles le Smart-hub sera exécuté. Pour rendre ce composant flexible pour ajouter et pour modifier les règles, nous proposons d'ajouter le concepts de plug-in à ce composant. Plus de détails seront expliqués dans le chapitre suivant.

3.3.1.5 Composant d'orchestration

Ce composant, Orchestration component en anglais, représente le coordinateur du Smart-hub (cf. figure 3.11). Il a deux rôles pour répondre à notre objectif F4 : d'une part, il coordonne la diffusion de données de DP vers les composants de Smart-hub pour l'acquisition de données (data acquisition - DA) selon les différentes options d'interaction de DP (cf. 2.1.1.2). De plus, il coordonne la diffusion de données des composants de Smart-hub vers les DC pour la génération de données (data generation - DG) selon les différentes options d'interaction. Autrement dit, il répond à la question : comment les données arrivent-elles à destination ? il ne faut pas confondre l'orchestration avec la communication, c'est l'orchestration qui demande la communication de faire des choses selon les modes d'interaction différents ; d'autre part, il gère l'interaction entre les composants de Smart-hub lui-même. Par exemple, l'interaction entre les composants de communication, le composant indépendant de domaine, etc.

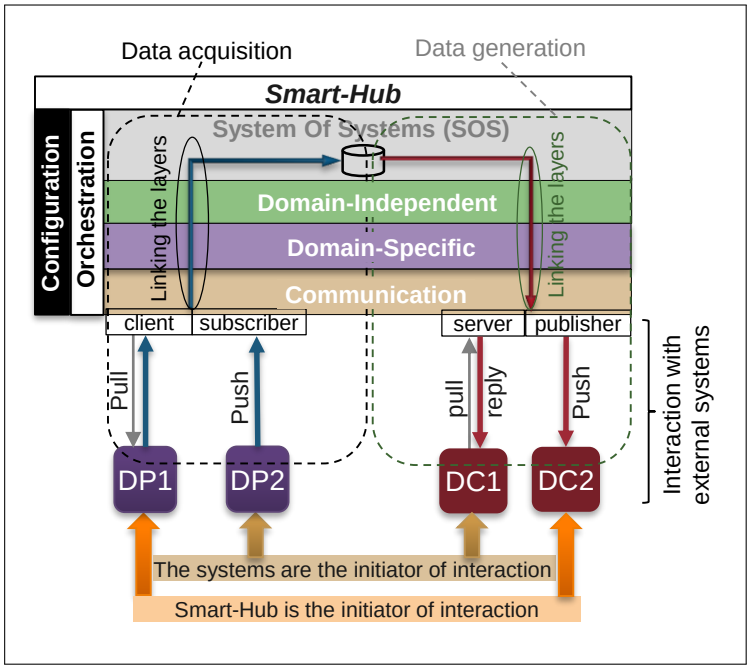


FIGURE 3.11 – Orchestration

Acquisition de données (data acquisition - DA)

D'une part, ce processus gère l'interaction du composant de communication avec les producteurs de données selon les options d'interaction pour récupérer les données. Nous avons classé les options suivantes pour l'interaction du Smart-hub avec les DPs.

- Pull Aperiodic : le Smart-hub est vu comme le client qui lance les demandes (request) de pull. Ce type de connexion est unicast car le Smart-hub interagit avec les DP directement.
- Pull Periodic : un cas particulier de l'option précédente. Ici, le Smart-hub lance les demandes (request) de pull périodiquement.
- Push Periodic/Aperiodic (PPA) : l'initiateur de données est le DP, donc le Smart-hub est vu comme un souscripteur qui s'abonne à ce DP et attend l'événement indiquant l'arrivée des données. La publication de données (périodiquement/apériodiquement, unicas/1-to-N) est une préoccupation du DP (qui est l'initiateur des données) et non du Smart-hub.

D'autre part, ce composant gère la liaison des composants horizontaux du bas vers le haut. Autrement dit, il définit la séquence d'exécutions de fonctions dans les composants en réception de données. Après, la réception de données (via push ou pull), le format de données est unifié sans toucher les sémantiques en utilisant les règles de conversion dans le composant spécifique au domaine. Ensuite, la sémantique est associée à un modèle pivot

via des règles de mapping dans le composant indépendant du domaine. Enfin, les données sont agrégées dans le référentiel central dans le composant du système de systèmes.

Génération de données (data generation - DG)

De la même manière que pour gérer l'acquisition de données, ce processus de génération de données a deux rôles : d'une part, il gère l'interaction du composant de communication avec les consommateurs de données selon les options d'interaction afin de fournir les données. Nous avons classé les options suivantes pour l'interaction des DC avec le Smart-hub :

- Pull Periodic/Aperiodic : le Smart-hub est vu comme le serveur qui écoute l'événement de demande (pull) de données de la part de DCs (des clients). En réception des demandes, le Smart-hub diffuse les données pour les DC. On note bien que le DC est l'initiateur de demande et sa connexion avec le Smart-hub est unicast. Donc, le Smart-hub ne se soucie pas si les demandes sont périodiques ou apériodiques, c'est la préoccupation du DC.

- Push APeriodic : le Smart-hub est vu comme le serveur qui initie la publication de données. Les données sont publiées a chaque fois qu'elles sont mises à jour. C'est le rôle du DCs de s'abonner à des données. En effet, nous pouvons avoir plusieurs consommateurs qui sont intéressés par des données, donc la connexion est 1-to-N.

- Push Periodic : un cas particulier de l'option précédente où les données sont publiées périodiquement.

D'autre part, ce processus relie les composants horizontaux dans le sens inverse du haut vers le bas. Il fournit les données de CDM à des interfaces de consommateur de données. D'abord, les données sont retirées au dépôt central et traitées pour générer les données dans le composant indépendant du domaine. Ensuite, les données sont affinées selon les règles de mapping pour générer les données dans la sémantique spécifique au domaine et avec un format unifié. Après, les données sont converties avec un format dans l'espace technique de l'interface de DC. Enfin, les données sont envoyées ou mises à disposition pour l'interface de consommateur de données pour être diffusées selon leurs options d'interaction.

3.3.1.6 Composant de configuration

Ce composant, Settings Component, gère la configuration du Smart-hub pour interagir avec les systèmes externes. Cela implique deux niveaux de configuration : la configuration spécifique au SmartHub et la configuration spécifique au système externe.

- **La configuration spécifique au SmartHub :** l'architecture contient différents composants avec des différents sous-composants tels que mécanismes de communication,

modèles spécifiques au domaine, règles de conversion, règles de mapping, etc. permettant d'exécuter les processus DA et DG. Pour chaque système externe, le Smart-hub a besoin d'utiliser une combinaison de ses sous composants afin d'exécuter le processus de DA ou DG. Donc, ce composant gère le paramétrage de Smart-hub pour utiliser les bons sous-composants pour interagir avec le système externe. La figure 3.12 illustre un exemple, nous configurons le Smart-hub pour interagir avec le producteur de données GMAO qui utilise un connecteur de système de fichiers, a un format tabulaire et possède une sémantique spécifique au système, etc.

- **La configuration spécifique au système externe :** les systèmes externes ont des paramètres à respecter tels que l'adresse IP, l'authentification, les données de filtrage (derniers jours, dernière semaine), etc. afin d'établir les interactions avec eux. Par exemple, pour interagir avec le GMAO, les données de paramètre sont (IP=xx.xx.xx.xx, un= "username", pwd="password"). Ces configurations font partie aussi du composant de configuration. Remarquer bien que les paramètres de configuration spécifique au système externe ne sont pas identiques pour tous les systèmes, par exemple, les systèmes utilisant les protocoles OPC UA utilisent les paramètres sécurité mode, depth, etc.

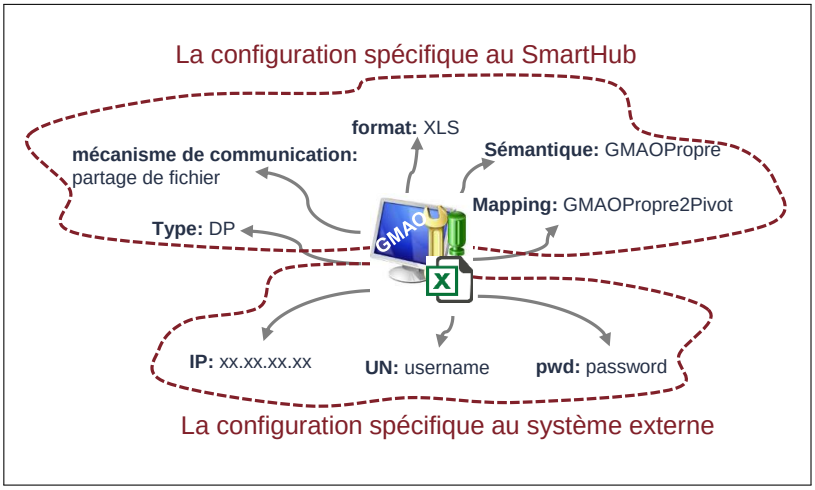


FIGURE 3.12 – Exemple de configuration de système GMAO

3.4 Conclusion

La mission d'échanger des données de façon interopérable nécessite de gérer plusieurs niveaux d'interopérabilité technique, syntaxique et sémantique. En fait, nous ne pouvons pas assurer un environnement où tous les systèmes suivent la même norme. D'autres solutions pour promouvoir l'interopérabilité comme les middlewares sont étroitement couplées et ne sont pas extensibles selon différents environnements. Pour cette raison, dans ce chapitre,

nous avons proposé une architecture modulaire et extensible qui permet de promouvoir l'interopérabilité entre les systèmes pour répondre à notre objectif F5. L'architecture s'appuie sur les concepts de génie logiciel de la séparation des préoccupations en utilisant les concepts de CBSE. Cela permet d'avoir des composants indépendants et réutilisables pour les différents niveaux d'interopérabilité en répondant aux objectifs F1, F2, F3 et F4. Grâce au CBSE, nous avons rendu notre architecture extensible en utilisant le concept de plug-ins. Cela nous permet d'étendre notre architecture à plusieurs composants selon l'environnement actuel et selon les différents niveaux d'interopérabilité à traiter. Pour conclure, ce chapitre a présenté la conception pour répondre a nos objectifs et dans le chapitre suivant, nous allons présenter les solutions techniques et l'implémentation de l'architecture proposée dans ce chapitre.

Chapitre 4

La solution technique et la mise en œuvre

Sommaire

Dans le chapitre précédent, nous avons proposé la conception de l'architecture. Ce chapitre présente la solution technique pour réaliser et mettre en œuvre l'architecture discutée dans le chapitre précédent afin de répondre à notre objectif F6. La première section est une présentation sur des outils et des techniques qui seront utilisés. Il introduit la plates-forme de développement Eclipse, le concept de plug-ins, le cadre de modélisation EMF, le langage de transformation ATL, et les outils de projections. La deuxième section présente la réalisation de l'architecture en utilisant les outils et techniques de la première section. L'implémentation de l'architecture est décomposée en trois briques : "la brique de l'interopérabilité et la modélisation" qui gèrent les problèmes de l'interopérabilité en implémentant les quatre composants horizontaux de l'architecture; "la brique d'orchestration" qui présente l'implémentation du composant d'orchestration; "la brique de configuration" qui présente l'implémentation du composant de configuration.

4.1 Outils et techniques

Les outils et techniques illustrés sur la figure 4.1 seront introduits dans cette section. D'abord, nous présentons la plate-forme "Eclipse" qui est un environnement de développement de logiciel. Elle est basée sur les concepts de plug-ins qui permettent d'étendre la plate-forme pour avoir de nouvelles fonctionnalités. Eclipse est une plate-forme indépendante des langages de programmation, cependant il est créé avec le langage de programmation Java qui sera utilisé pour réaliser notre architecture. Ensuite, nous présentons le cadre de modélisation EMF qui est l'un des plug-ins sur Eclipse. Puis, nous discutons les techniques permettant de transformer des modèles parmi lesquelles ATL a été choisi. Enfin, nous présentons les techniques pour la projection basée sur Java, EMF et Eclipse.

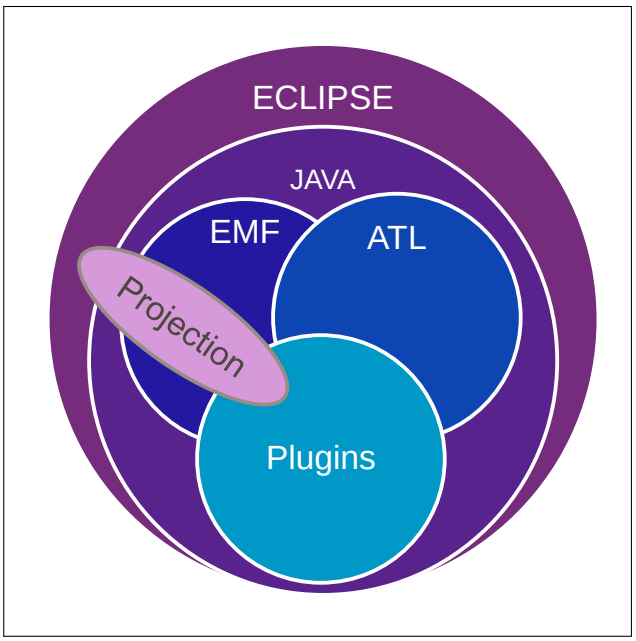


FIGURE 4.1 – Les outils pour implémenter l'architecture

4.1.1 Plateforme Eclipse et plug-ins

Eclipse est un environnement de développement de logiciel libre pour le développement d'outils hautement intégrés. La plate-forme Eclipse, qui fait partie du projet Eclipse, est un cadre extensible et ouvert pour construire des environnements de développement intégré (IDE) qui sont utilisés pour créer des applications diverses. En fait, il définit simplement la structure de base d'un IDE dans lequel des outils spécifiques sont branchés pour créer un IDE particulier.

En effet, la plate-forme Eclipse devient une fondation d'applications génériques basées sur les plug-ins. La plate-forme Eclipse elle-même est composée de plug-ins qui permettent aux autres plug-ins de s'étendre. Chaque plug-in apporte des fonctionnalités qui peuvent

être invoquées par l'utilisateur ou réutilisées et étendues par d'autres plug-ins. Les plug-ins eclipse fournissent des extensions pour définir des fonctionnalités qui peuvent être entendues via des plug-ins en les définissant comme des points d'extension (extension points). Les principaux plug-ins d'Eclipse sont écrits en Java. Cependant, Eclipse est langue neutre. Il peut supporter divers langages de programmation (par exemple C/C++, Cobol, PHP, etc.) et d'autres aspects du développement (base de données, conception avec UML ...) ajoutés en tant que plug-ins. Le développement, qui est basé sur les plug-ins, apporte des avantages techniques et économiques. Techniquement, il permet la création de composants modulaires et faiblement couplés. L'application de base et les plug-ins peuvent être réalisés séparément, donc les développeurs peuvent travailler plus indépendamment. Économiquement, la modularité permet de créer des composants en parallèle et les réutiliser ce qui entraîne une réduction de la période de livrable et du coût de développement. Comme nous avons déjà vu dans la proposition (cf. chapitre 3), notre architecture repose sur des composants extensibles. Donc, ce paradigme de développement basé sur les plug-ins jouera un rôle important dans le développement de notre architecture.

4.1.2 Eclipse Modeling Framwork

L'ingénierie basée sur les modèles est appliquée souvent dans l'industrie et le milieu universitaire. Cela a plusieurs avantages : l'amélioration de la productivité du logiciel; la modélisation abstraite facilite la compréhension commune entre les experts du domaine et les développeurs; l'implémentation de notre architecture repose sur les aspects de modeling. Cela soulève la question : quels sont les outils de modeling? L'un des cadres de modeling est "Eclipse Modeling Framework". Il est composé des divers outils ajoutés comme plug-ins audessus d'Eclipse. EMF implémente les normes de MDA. En théorie, EMF et MDA sont alignés et considérés comme le même espace technique [Bézivin, 2006]

EMF fournit une approche simple et pragmatique pour modéliser et générer le code pour développer des outils et des applications. Les modèles peuvent être définis via des approches différentes JAVA, XML, ou UML diagramme de classes à partir de laquelle vous pouvez ensuite générer les autres approches. Par exemple, à partir de diagramme de classes UML, nous pouvons générer les autres représentations. Autrement dit, EMF est une projection entre des espaces techniques : modeling (UML diagramme de classes), Java et XML. En EMF, les metamodèles sont exprimés via un metametamodèle appelé "Ecore" (niveau M3). Ecore correspond à peu près à une partie de norme MOF de OMG. [Steinberg et al., 2008]. Les détails techniques d'ECORE ne font pas partie de l'objectif de ce travail. De plus, EMF est composé de plusieurs outils pour gérer, accéder et sérialiser efficacement des (méta)modèles. Les (méta)modèles sont sérialisés en XMI qui est une norme pour la sérialisation des métadonnées en utilisant XML.

4.1.3 Transformation de modèles

La transformation de modèles est le processus de production d'un ou plusieurs modèles de sortie (Model B) de l'un ou de plusieurs modèles d'entrée (Model A) tout en conservant la conformité avec le métamodèle (MetaModel A, MetaModel B). La transformation de modèle peut être atteinte par différents langages de transformation. Nous pouvons utiliser les langages généralistes qui s'appuient directement sur la représentation abstraite du modèle comme l'interface de programmation (API) de EMF. Dans ce cas, le programmeur doit écrire le code de mapping entre les modèles pour chercher les éléments dans les modèles, écrire les règles de transformation, créer les éléments cible, spécifier l'ordre d'exécution, etc. Cependant, il existe des langages de transformation qui sont dédiés à la transformation de modèles. Ces langages gèrent la transformation de façon plus abstraite comme ATLAS Transformation Language (ATL) [Jouault et al., 2008, Jouault and Kurtev, 2006], QVT, etc. ATL est un langage hybride des constructions déclaratives et impératives. Il permet de définir une transformation de modèle à modèle sous la forme d'un ensemble de règles [Combemale, 2008]. ATL utilise des règles simples qui peuvent être comprises ou écrites par des experts du domaine. Selon [Van Amstel et al., 2011], ATL, qui est bien soutenu par Eclipse et EMF, fournit de meilleures statistiques de performance par rapport à d'autres moteurs de transformation tels que QVT. Donc, nous sélectionnons ATL pour gérer l'interopérabilité sémantique entre le composant spécifique au domaine et le composant indépendant du domaine dans notre implémentation. Il existe une IDE pour ATL au-dessus d'Eclipse pour faciliter le travail de développeur. Il fournit beaucoup de fonctionnalités pour compiler, lancer et exécuter les règles de transformation.

4.1.4 Mécanismes de projections

L'interaction entre l'espace de modélisation et l'autre espace technique peut être réalisée par des projecteurs. Nous avons proposé de mettre en œuvre notre architecture dans l'espace de modélisation EMF. Les données de DP et DC sont dans un autre espace technique différent de l'espace de modélisation EMF. Par conséquent, certaines techniques de projection doivent être prises en considération. Les outils de projections sont catégorisés en "injecteurs" pour le pont d'un espace technique (java, bases de données, XML, fichiers plats, etc.) vers l'espace technique EMF et "extracteurs" pour le pont vice versa.

Un certain nombre de ponts prédéfinis d'injecteurs et de projecteurs existent tels que Model2SQL, Model2XML, XML2Model, Model2Text, etc. Les formats de données XML ou grammaticaux peuvent être projetés par des outils logiciels en utilisant des règles simples [Efftinge and Völter, 2006]. De plus, des ponts ad-hoc spécifiques peuvent être ajoutés en utilisant un langage de programmation généraliste tel que Java.

4.2 Le cadre Smart-hub : l'implémentation de l'architecture

Le chapitre précédent a proposé une solution conceptuelle qui doit être réalisée. Donc, cette section présente la solution technique et la mise en ouvre pour l'architecture proposée afin de répondre à notre objectif F6. Dans ce manuscrit, nous nous référons à la solution technique par cadre ou système de façon interchangeable.

4.2.1 Vue globale des briques d'implémentation

Nous décomposons la mise en œuvre de l'architecture en trois briques tel qu'illustrées sur la figure 4.2. La première brique "Brique de l'interopérabilité et la modélisation" décrit l'implémentation des quatre composants horizontaux de l'architecture (composant de communication, composant spécifique au domaine, composant indépendant du domaine et composant du système de systèmes) qui gèrent les problèmes de chaque niveau de l'interopérabilité. Le deuxième brique "Brique d'orchestration" présente l'implémentation du composant vertical "composant d'orchestration" qui gère l'acquisition et la génération de données en temps réel entre les couches et les systèmes externes. Enfin, la dernier brique "Brique de configuration de Smart-hub" introduit la mise en œuvre du composant vertical "composant de la configuration".

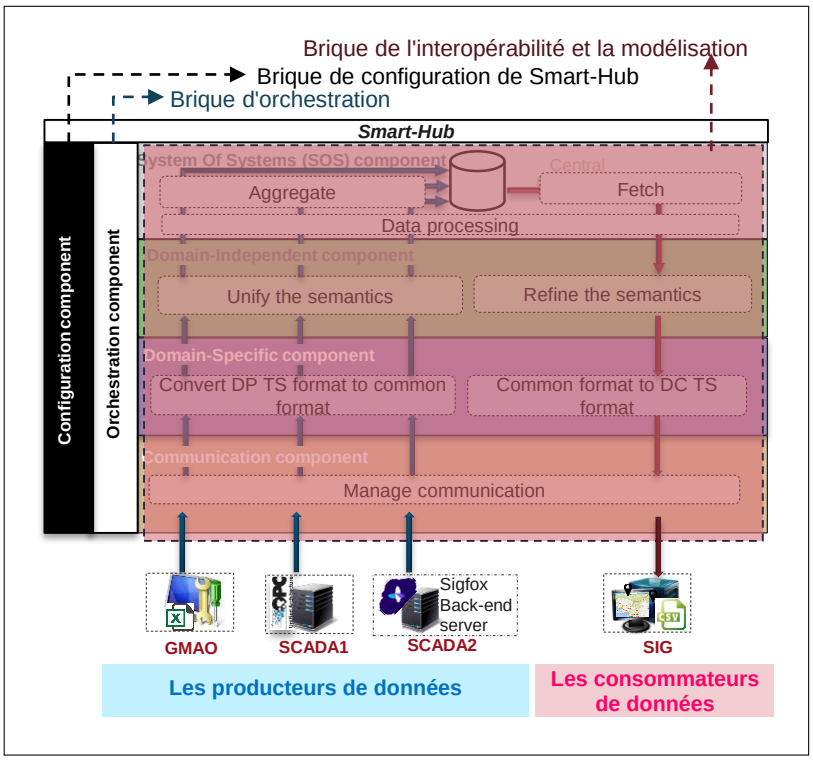


FIGURE 4.2 – Vue globale des briques d'implémentation

4.2.2 Implémentation de la brique de l'interopérabilité et la modélisation

Dans cette section, nous présentons les détails techniques pour réaliser chaque composant horizontal qui gère les différents niveaux de l'interopérabilité. Pour chaque composant, nous définissons les différentes fonctionnalités fournies et les interfaces pour l'étendre.

4.2.2.1 Composant de communication

Le composant de communication doit fournir les fonctionnalités suivantes : établir une communication avec le système externe ; déconnecter du système externe ; recevoir des données (et/ou méta données) du producteur de données ; envoyer des données (et/ou méta données) au consommateur de données. Ces fonctionnalités sont définies en tant que point d'extensions pour être étendues par différents mécanismes de communications (connecteur client OPC UA , connecteur fichiers, connecteur Rest Webservice). Chaque mécanisme est créé en tant que plug-in d'extension identifié qui implémente et contribue aux fonctionnalités telles que représentées sur la figure 4.3. Le regroupement de ces mécanismes de communication peut être considéré comme un référentiel (ou catalogue) de connecteur à partir duquel le développeur utilise celui qui convient pour le système connecté.

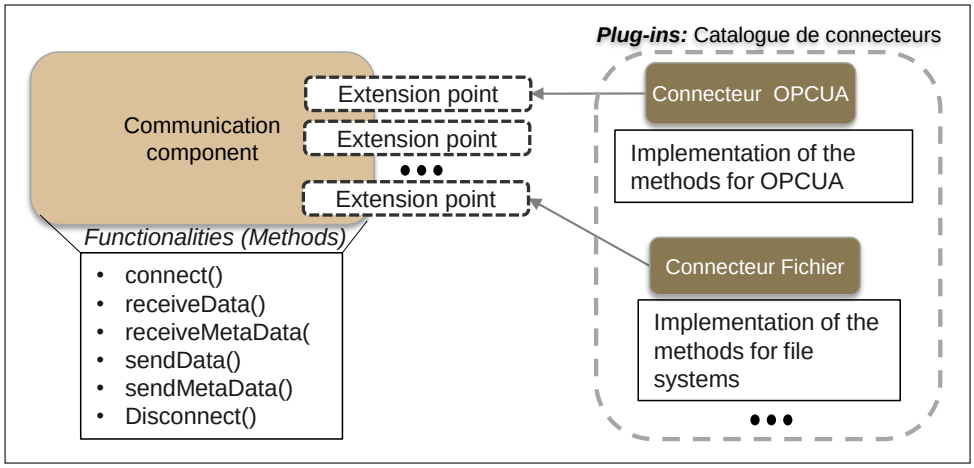


FIGURE 4.3 – Exemple d'extension du composant de communication via plug-ins

4.2.2.2 Composant spécifique au domaine

Le composant spécifique au domaine gère la projection entre le format de données dans l'espace technique du système et l'espace de modélisation EMF. Il définit deux fonctions : *ProjectToModel* qui se comportent comme un injecteur et *ProjectFromModel* qui se comportent comme un extracteur. Ces règles sont gérées dans un référentiel (catalogue) "Conversion Rules Functional Repository".

Grace au plug-in, nous pouvons ajouter diverses règles de projection qui implémentent et contribuent aux fonctions (ProjectToModel et/ou ProjectFromModel). Comme nous

sommes dans l'espace de modélisation, les métamodèles spécifiques à la sémantique du système/domaine sont un prérequis à la fonctionnalité des règles de projection. Ces méta-modèles spécifiques au domaine sont définis à l'aide du métamodèle Ecore et gérés dans un référentiel "Common Format Repository". Ils peuvent être partagés par différents systèmes, soit parce qu'il s'agit d'un format générique (tel que le modèle de données OPC UA), soit parce que le développeur peut regrouper des systèmes qui fournissent les mêmes données spécifiques au domaine. Par exemple, le schéma OPC UA est définit via XSD, donc, grâce au outils de projection XSD2Ecore, nous pouvons facilement générer le metamodèle OPC UA. Les données d'un fichier Excel et CSV sont les données en tableau, donc nous avons défini un meta-modèle "Tabular MetaModel" dans le référentiel pour gérer les données structurées en lignes et colonnes. Le système SCADA2 utilise un format spécifique, alors nous avons défini un metamodèle spécifique au système "SCADA2 MetaModel" dans le référentiel.

Les plug-ins utilisent ces metamodèles afin de créer les projecteurs. La figure 4.4 illustre les listes de plug-ins pour gérer les formats de systèmes dans notre exemple fil rouge : Excel-Projection et CSVProjection utilisent TabularMetaModel pour gérer les formats de systèmes GMAO et SIG respectivement ; OPUCAProjection utilise "OPC UA MetaModel" pour gérer les formats de systèmes SCADA1 ; SCADA2Projection utilise "SCADA2 MetaModel" pour gérer les formats de systèmes SCADA2. La figure 4.5 montre un exemple de projection (injecteur) pour les données de système SCADA1. Une fois les données reçues via le connecteur OPC UA, elles sont projetées vers un modèle conforme au métamodèle générique OPC UA .

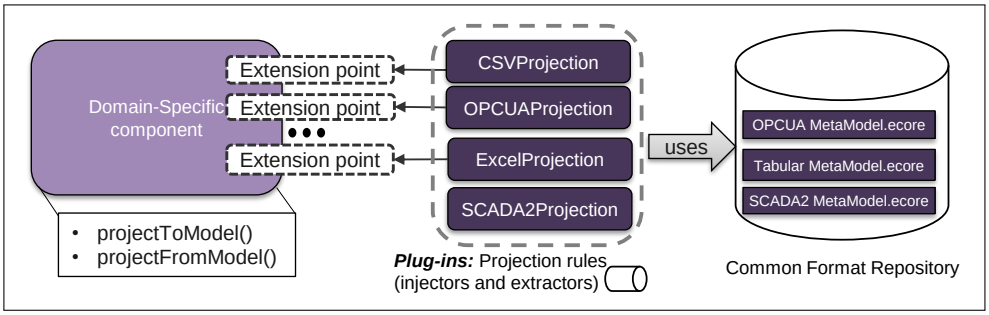


FIGURE 4.4 – Exemple d'extension du composant spécifique au domaine via plug-ins

4.2.2.3 Composant indépendant du domaine

Le composant précédent nous fournit une syntaxe unifiée pour les données. C'est le rôle de cette composante de gérer l'unification de la sémantique. Il s'appuie sur un métamodèle qui fournit une sémantique unifiée et des règles de mapping pour unifier et affiner les séman-tiques de données.

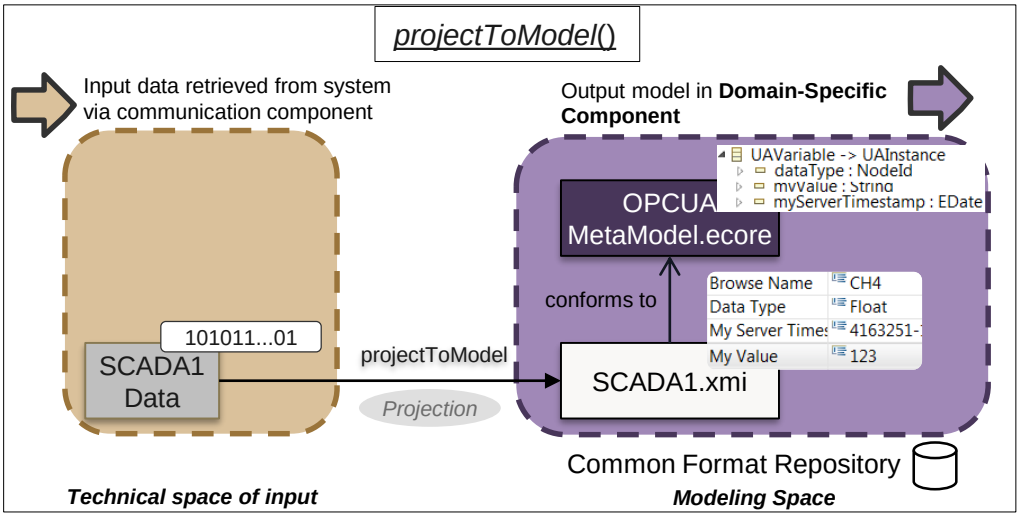


FIGURE 4.5 – Exemple de projection de données de système SCADA1

La sémantique unifiée

Afin de gérer les sémantiques, nous proposons la définition d'un méta-modèle **Common Data Model (CDM)** à un niveau abstrait et agnostique. Ce modèle, qui est inspiré de [Mahnke et al., 2009, Nativi et al., 2008, Robert et al., 2016], est générique et indépendant des domaines spécifiques tel qu'illustré dans figure 4.6. Il est défini par le modèle "Ecore". Les principaux concepts sont :

- **Objet** : un conteneur identifié de variables de données et d'objets imbriqués qui sont organisés en arborescence hiérarchique. L'identificateur d'objet racine doit être unique pour un système particulier.
- **Data Variable** : est un conteneur de données caractérisées par un nom, un type de données et une valeur. Il peut inclure des variables de données imbriquées et des méta-variables.
- **Meta Variable** : est un conteneur de données qui fournit des informations supplémentaires pour une variable de données telle qu'une unité. Ces données sont fixes, et dans l'état normal d'une application, il ne change pas sa valeur. Il ne détient aucune hiérarchie au-delà.
- **Value** : est utilisé pour maintenir la valeur avec un horodatage optionnel, c'est-à-dire le moment de l'acquisition des données par le Smart-hub.
- **DataType** : cette version de modèle unifié gère les types de données simples, mais elle n'empêche pas de l'étendre pour gérer les types complexes comme les tableaux.

Ce metamodèle est géré dans un référentiel que nous appelons "Intermediate Repository".

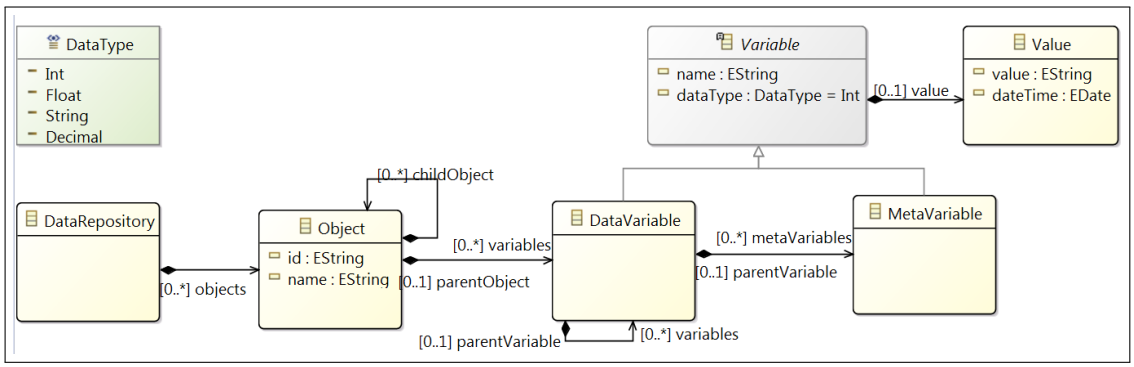


FIGURE 4.6 – Modèle de données "Common Data Model"

Les règles de conversion

Les règles de conversion ont pour but d'étendre l'interopérabilité sémantique en associant des metamodèles spécifiques au domaine avec ce métamodèle indépendant du domaine (unifier - unify) et vice versa (raffiner - refine). Pour chaque métamodèle spécifique au domaine, un plug-in est ajouté pour contribuer à ces fonctionnalités. Là encore, les technologies basées sur des modèles facilitent le développement. Comme nous sommes dans l'espace de modélisation, nous nous référons aux règles de mapping à des réglés de transformation. Ces règles sont stockées et gérées dans un référentiel "Transformation Repository". Nos transformations actuelles sont principalement basées sur l'outil ATL [Jouault et al., 2008, Jouault and Kurtev, 2006]. La figure 4.7 illustre des exemples de plug-ins ATL pour unifier/affiner les données entre les modèles spécifiques de domaine comme le modèle OPC UA , le modèle Tabular, le modèle_SCADA2.

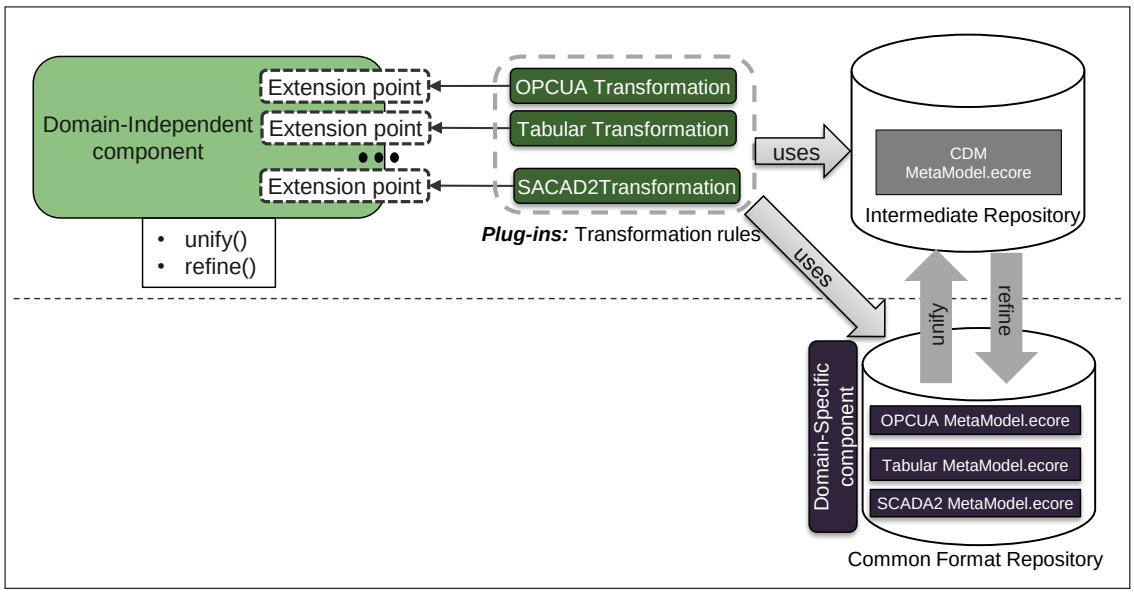


FIGURE 4.7 – Exemple d'extension du composant indépendant du domaine via plug-ins

La figure 4.8 montre un exemple de transformation (unify) de modèle de système GASS, qui est conforme au métamodèle OPC UA , vers un modèle conforme au CDM. En fait, cette transformation implique un modèle de données complémentaire du composant de la configuration . Il contient des informations sur le système connecté défini dans le composant de paramétrage comme l'identification de système. Plus de détails seront expliqués plus tard.

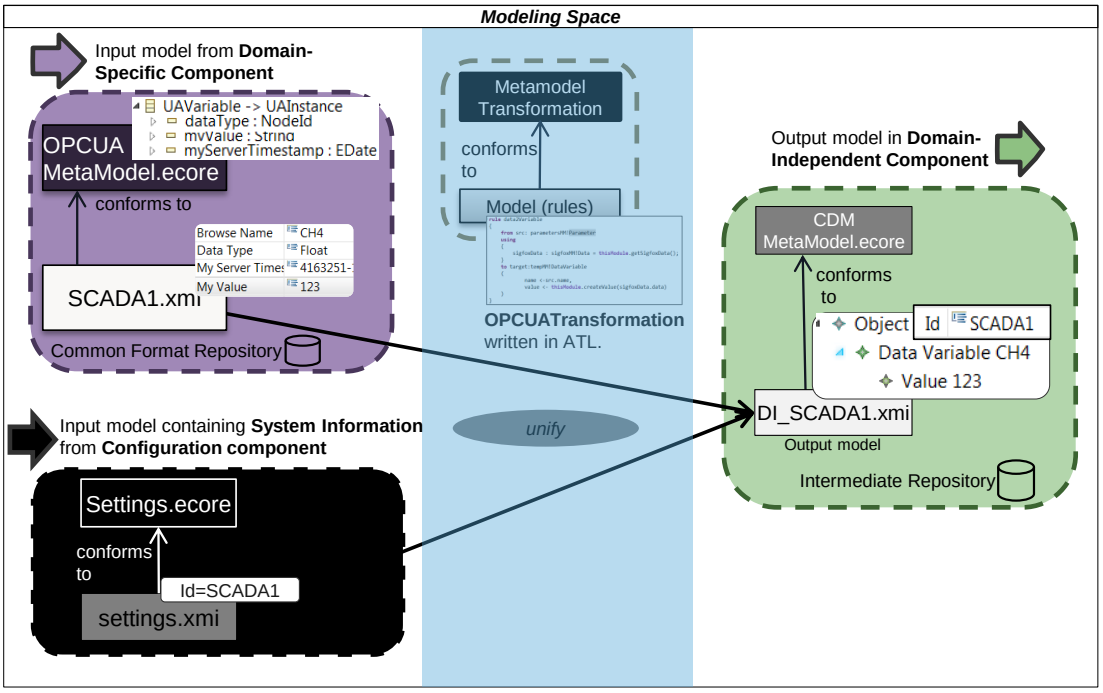


FIGURE 4.8 – Exemple de l'unification de la sémantique de modèle de système SCADA1 vers CDM

4.2.2.4 Composant du système de systèmes

Ce composant, qui utilise le métamodèle CDM, contient le référentiel central et le dépôt de données. Les données sont agrégées dans ce référentiel puis sont retirées de celui-ci. De plus, il contient les règles métiers/opérationnelles pour traiter les données. Nous proposons l'utilisation de l'approche post-agrégation pour traiter les données. En fait, les données des systèmes producteurs de données (DP) peuvent être partagées par plusieurs règles requises par différents consommateurs de données (DC). Comme les données de DC sont construites à partir des données de DP, c'est plus pratique d'avoir les données originales des producteurs de données agrégées dans le CDM. Les implémentations d'opérations sont données ci-dessous :

Aggregate

Cette opération "créer" ou "mettre à jour" le référentiel central avec les données de la couche indépendante du domaine. Chaque instance de système a un objet correspondant dans le

référentiel, si le CDM reçoit les données du système pour la première fois, alors l'objet du système est créé dans le référentiel. Sinon, l'objet préexistant est mis à jour par les nouvelles valeurs reçues. Cette opération n'est pas spécifique au système et est donc implémentée de manière générique dans le cœur du Smart-hub. La figure 4.9 illustre un exemple d'agrégation pour le système SCADA1. Nous supposons qu'il y a déjà des données de SCADA1 et SCADA2. Ensuite, les données sont agrégées pour faire la mise à jour de données pour SCADA1.

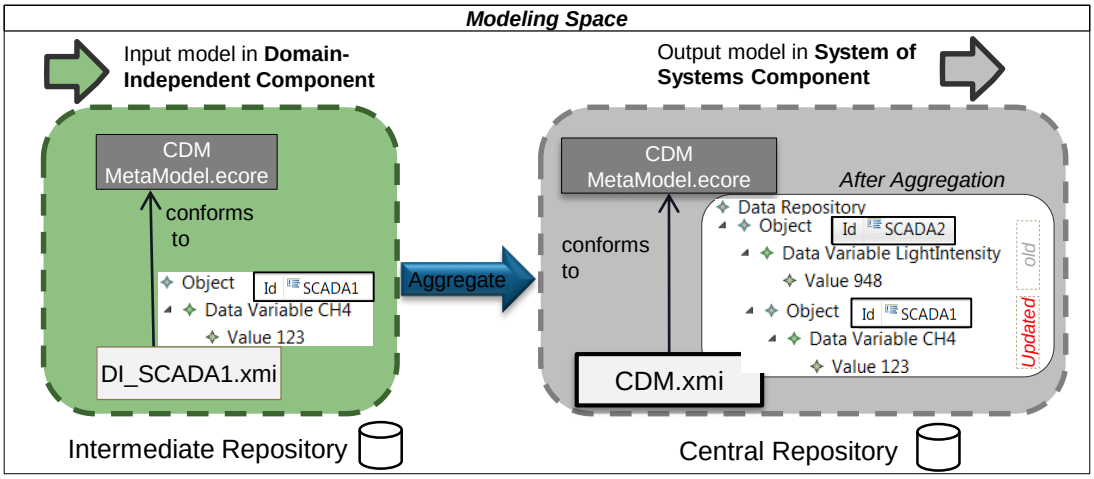


FIGURE 4.9 – L'agrégation des données de SCADA1

FetchAndProcess

En correspondance avec l'approche de traitement (post-agrégation), nous proposons de combiner l'implémentation des phases de retrait et de traitement les données. Autrement dit, nous retirons les données à partir du référentiel CDM, effectuons les règles métiers/opérationnelles requises par le DC et générons un modèle pour le composant indépendant du domaine. Encore une fois, les technologies basées sur des modèles comme ATL facilitent le chargement de modèle pour retirer/chercher les données et la rédaction de ces règles métiers/opérationnelles par des experts du domaine.

Pour chaque différent modèle spécifique au domaine utilisé par des DC, un plug-in d'extension est implémenté pour contribuer au retrait et le traitement de données. Dans notre exemple fil rouge, nous avons un seul DC alors nous avons ajouté un seul plug-in pour créer le modèle destiné au DC SIG tel qu'illustré dans la figure 4.10. Ce plug-in a pour but de traiter les données agrégées à partir de (GMAO, SCADA1 et SCADA2) dans le référentiel central. Les détails de transformation de ce plug-in sont illustrés dans figure 4.11.

La règle de transformation pour le système SIG charge d'abord les données de SCADA1, SCADA2, et GMAO. Ensuite, ces données sont traitées. Le traitement implique d'une part l'association de chaque système SCADA1 et SCADA2 avec leurs propres coordonnées, et d'autre part la modification de l'interopérabilité sémantique des noms de coordonnées. En effet,

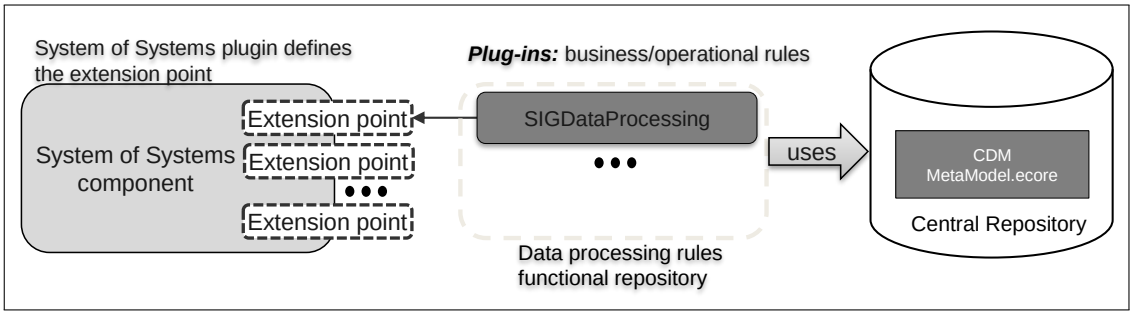


FIGURE 4.10 – Exemple d'extension du composant système de systèmes via plug-ins pour le traitement de données

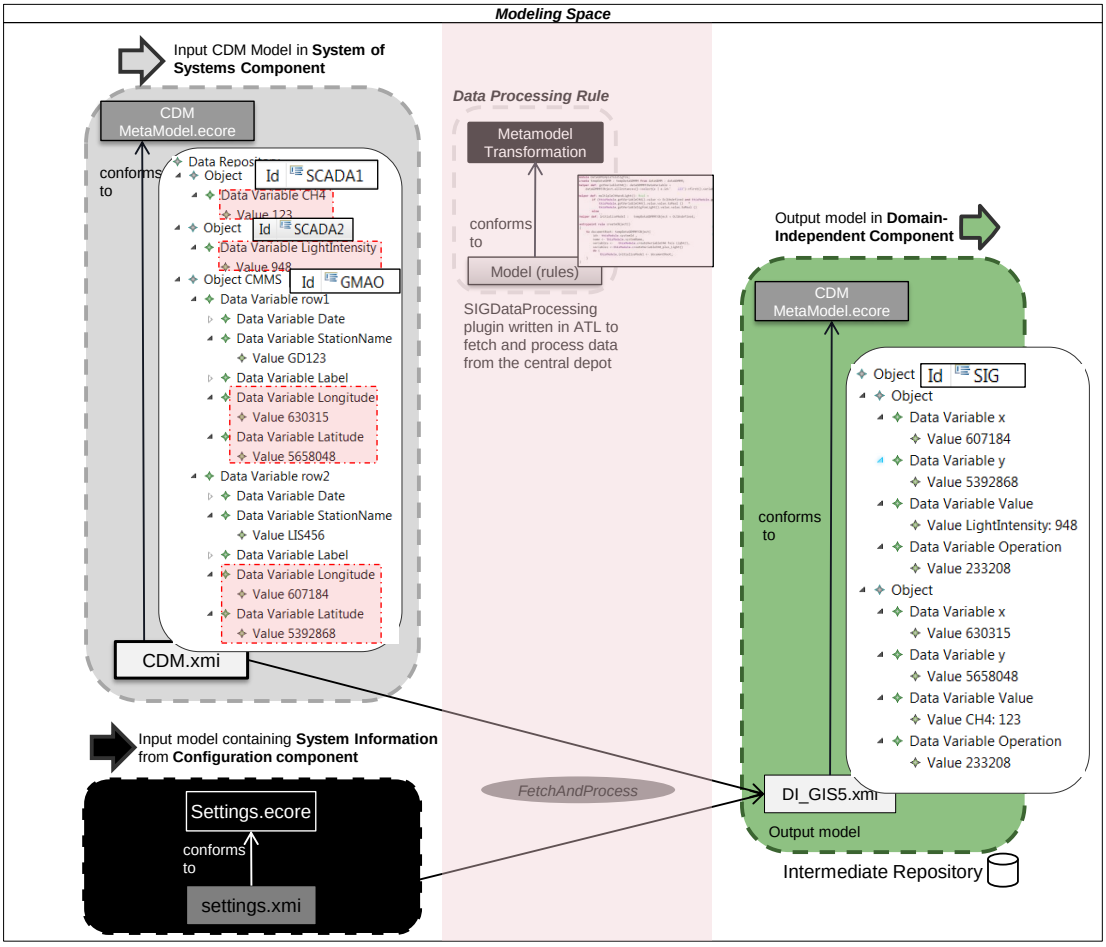


FIGURE 4.11 – Exemple de traitement des données destinées au système SIG

le système GMAO utilise "longitude" et "latitude" pour les noms de coordonnées, mais le système SIG utilise "X" et "Y". Enfin, ce modèle généré suit alors le processus inverse précédemment décrit : elle est raffinée jusqu'à la couche spécifique au domaine et finalement est fournie au système consommateur en utilisant son mécanisme de communication.

4.2.3 Implémentation de la brique d'orchestration

Ce bloc introduit la mise en œuvre du composant d'orchestration qui gère la réception des données provenant des producteurs de données vers le Smart-hub (appelé acquisition de données - DA) et les envoie du Smart-hub aux consommateurs de données (appelé génération de données - DG) selon différentes options d'interaction. Dans la section suivante, nous présentons d'abord l'implémentation de processus DA et ensuite l'implémentation de processus DG.

4.2.3.1 Acquisition de données (data acquisition - DA)

L'acquisition de données a deux rôles comme illustré dans figure 4.12 : DAExecution et DAInteraction.

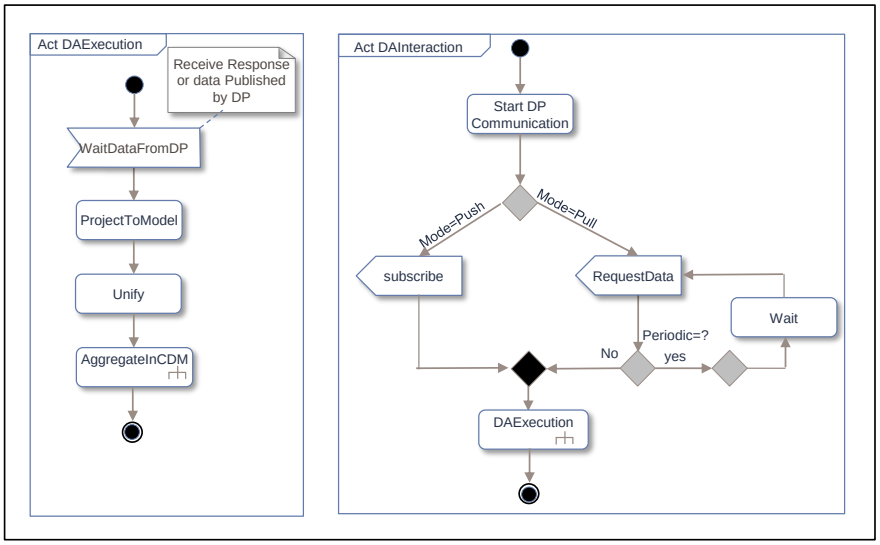


FIGURE 4.12 – Les figures UML des activités d'orchestration pour l'acquisition de données

DAExecution

le DAExecution fait la liaison entre les fonctions des couches horizontales du bas vers le haut pour aggreger les données dans le CDM tel qu'illustré dans le partie gauche de la figure 4.12. D'abord, le Smart-hub attend que la couche de communication reçoive des données de la part des DPs. Ensuite, les données sont projetées vers un modèle dans le composant spécifique au domaine. Ensuite, le modèle est unifié vers un modèle conforme au CDM dans le composant indépendant du domaine. Enfin, le modèle est agrégé dans le dépôt central "composant du système de systèmes".

DAInteraction

Le DAInteraction gere les différents options d'interactions du Smart-hub avec des DP comme illustré dans la partie droite de figure 4.12. Après avoir lancé la communication, si l'option d'interaction de DP est "Push", le Smart-hub s'abonne sur les données de DP. Dès que les DP publient les données (périodiquement ou apériodiquement), le processus de DAInteraciton est déclenché. Sinon, si l'option d'interaction de DP est "Pull", le Smart-hub demande les données de DP (périodiquement ou apériodiquement). Dès que le DP répond à la demande, le processus de DAInteraciton est déclenché.

4.2.3.2 Génération de données

Le processus de génération de données implique deux rôles : DGExecution et DGInteraction tels qu'illustrés dans le figure 4.13

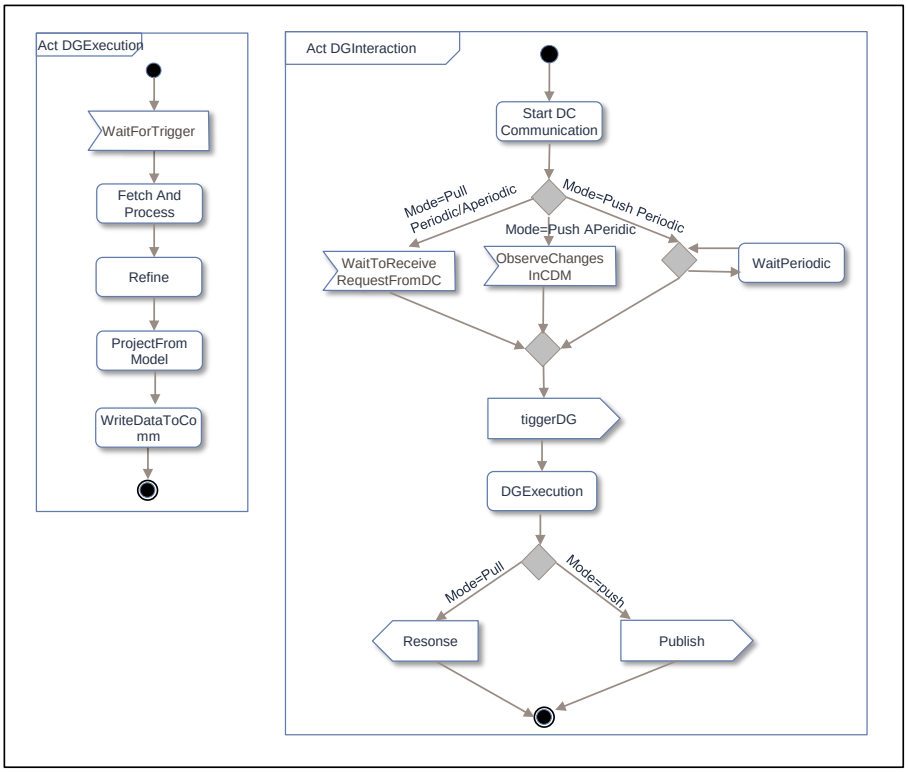


FIGURE 4.13 – Les figures UML des activités d'orchestration pour la génération de données

DGExecution

Le processus de DGExecution gère la liaison des composants du haut vers le bas afin d'alimenter le composant de communication avec les données de CDM. Ce processus attend une commande de déclenchement comme illustré dans la partie gauche de la figure 4.13.

Dès qu'il reçoit une commande, les données sont retirées du dépôt central et traitées pour générer un modèle dans le composant indépendant du domaine. Ensuite, le DGManager demande d'affiner ce modèle pour générer un modèle spécifique au domaine. Enfin, le modèle est extrait pour générer les données dans l'espace technique de système et d'alimenter le composant de communication.

DGInteraction

Le DGInteraction gère l'interaction du Smart-hub avec les DC selon leurs différentes options d'interactions comme illustré dans la partie droite de la figure d'activity. Selon les différentes options, le Smart-hub met à disposition ses données pour alimenter le composant de communication. Si les DCs sont basés sur l'options "Push", il faut que le Smart-hub lance la diffusion de données en alimentant le composant de communication. Alors, si l'option est de type "Push apériodique", le Smart-hub déclenche le processus de DGExecution quand il y a une changement dans le CDM. Si l'option d'interaciton est de type "Push periodique", le Smart-hub déclenche le processus de DGExecution périodiquement. Avec la dernière option "Pull" (soit apériodique ou périodique), les demandes sont lancées par le DC. Alors, le processus de DGExecution est déclenché à chaque fois que le Smart-hub reçoit une demande (pull apériodique ou périodique).

Le DG et le DA sont tous les deux mis en œuvre en utilisant le langage de programmation Java. Les systèmes basés sur les options d'interaction "Pull" sont mis en œuvre en utilisant le concept de l'appel de méthode tandis que les options d'interaction "Push" sont mis en œuvre en utilisant le modèle "ObserverPatthern" [Gamma et al., 1995]. Plus des détails sur ce modèle sont disponibles dans [Gamma et al., 1995]. Dans ce travail, nous avons seulement testé et mis en ouvre les options Pull Périodique et nous avons adapté l'option Push Périodique en utilisant les concepts d'appel de méthodes.

4.2.3.3 Programmation concurrente

Comme il s'agit d'un environnement en temps réel, les processus d'acquisition de données et de génération de données doivent être exécutés simultanément. La programmation concurrente est un paradigme de programmation dans lequel plusieurs calculs sont exécutés en même temps. Dans le Smart-hub, ces processus DA et DG doivent être exécutés en même temps avec une communication asynchrone. Par exemple, si le Smart-hub lance une demande auprès de DP, le Smart-hub n'a pas besoin d'attendre une réponse. Il peut effectuer d'autres tâches simultanément jusqu'à ce qu'une réponse soit disponible. Les avantages et les inconvénients de la communication synchrone ou asynchrone sont hors de portée de notre recherche.

Alors que la concurrence est répandue, elle soulève également des problèmes intéressants tels que la gestion des ressources partagées, la synchronisation, les conditions de

concurrence, les interblocages, etc. Le Smart-hub est confronté au problème de la gestion des ressources partagées. Par exemple, le dépôt central CDM est une ressource partagée qui est mise à jour simultanément avec les données des DPs. De ce fait, il y a un risque de perte de données si plusieurs DPs mettent à jour le CDM de manière simultanée. À l'inverse, le CDM est la source de données pour les consommateurs de données, il faut donc accéder simultanément au CDM pour générer des données pour les consommateurs de données. Le contrôle de la concurrence a été activement étudié depuis plusieurs années et de nombreuses solutions existent [Andrews, 1991, Magee and Kramer, 1999].

Notre problème correspond au problème de synchronisation bien connu Reader/Writer. Dans ce problème, les lecteurs (Readers) et les écrivains (Writers) partagent une ressource commune. Les lecteurs exécutent la transaction pour lire les données de la ressource partagée tandis que les écrivains exécutent la transaction pour écrire des données dans la même ressource partagée. La solution la plus appropriée dans notre contexte est d'utiliser une technique de programmation générale appelée "passing the baton" [Andrews, 1991]. Dans cette technique, un écrivain nécessite un accès exclusif, mais n'importe quel nombre de lecteurs peut lire simultanément [Magee and Kramer, 1999]. Ainsi, dans le Smart-hub, un processus de génération de données multiples peut lire des données simultanément par plusieurs threads tant qu'aucun thread de DA n'écrit exclusivement sur le CDM tel qu'illustré dans figure 4.14. Grâce à l'API "Java Concurrency", nous pouvons résoudre ce problème de concurrence en utilisant spécifiquement l'interface ReadWriteInterface [Lea, 1999, Magee and Kramer, 1999]. Cette interface utilise une paire de verrous pour l'accès en lecture et en écriture. Ainsi, le verrouillage de lecture peut être maintenu simultanément par plusieurs threads DG tant que les threads ne conservent pas le verrou d'écriture pour DA.

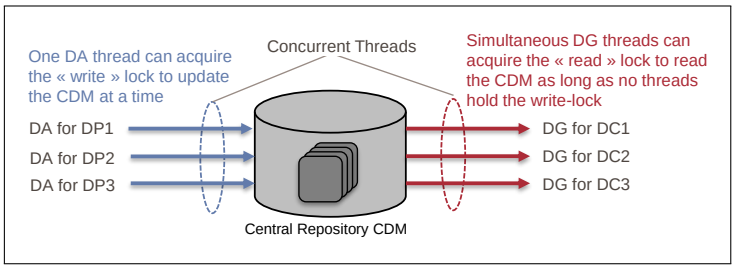


FIGURE 4.14 – La simultanéité des processus de DA et DG

4.2.4 Implémentation de la brique de configuration

Ce bloc introduit la mise en œuvre du composant de configuration de Smart-hub. Jusqu'au moment, nous avons présenté la mise en œuvre d'un certain nombre de composants, mais comment nous commandons au Smart-hub de charger le bon plug-in pour interagir avec les systèmes externes? Par exemple : charger le plug-in OPC UA au niveau du composant de communication pour interagir avec le système SCADA1 ; le plug-in TabularTransdorma-

tion au niveau de composant indépendant du domaine pour unifier les données de système GMAO, etc. Est que le système est un DP et un DC? Quel est le mode d'orchestration du système? Comment ajouter les paramètres d'interaction de chaque système, par exemple l'adresse IP, données de l'authentification, etc. La réponse à ces questions est le rôle du composant de paramétrage. Nous avons alors défini un référentiel de paramétrage (Setting repository) afin de gérer toutes les configurations du Smart-hub. Nous avons divisé cette configuration en deux niveaux : la configuration spécifique au Smart-hub et la configuration spécifique aux systèmes externes. La première traite le chargement des plug-ins pour chaque système et le choix de mode d'orchestration, la seconde gère les paramètres pour interagir avec ces systèmes externes.

4.2.4.1 La configuration spécifique au Smart-hub

Ces configurations servent à spécifier le type de systèmes externes (DP ou DC) avec son mode d'orchestration et à charger les plug-ins (de connecteur pour le composant de communication, de projections pour le composant spécifique au domaine, d'unification pour le composant indépendant du domaine, etc.) pour interagir avec ces systèmes externes comme illustré dans la table 4.1. Par exemple pour agréger les données à partir de DP SCADA1, le Smart-hub doit être configuré pour utiliser OPC UA pour le connecteur, OPC UA Projection pour les règles de projection, OPC UA Transformation pour les règles d'unification et le mode "pull périodique" au niveau d'orchestration. De la même façon pour générer les données pour le DC SIG, le Smart-hub doit être configuré pour utiliser FileSystem pour le connecteur, CSV-Projection pour les règles de projection, etc.

Role	System	Load plugins				Orchestration Mode
		Comunication	Domain Specific	Domain Independent	FetchAndProcess	
DP	SCADA1	OPCUA	OPCUAProjection	OPCUATransformation	-	Pull Périodique
DC	SIG	FileSystem	CSVProjection	TabularTransformation	SIGDataProcessing	Push Périodique

TABLEAU 4.1 – Exemple de configuration spécifique au Smart-hub pour interagir avec SCADA1 et SIG

Cependant, il y a plusieurs systèmes distribués identiques qui partagent le même mécanisme de communication, le même format de données et la même sémantique dans l'environnement industriel. Par exemple, dans le réseau de gaz (cf. figure 4.15), des centaines de systèmes SCADA d'analyseurs de gaz basé OPC UA (SCADA1, SCADA2 et SCADA3, etc.) identiques sont distribués sur plusieurs sites. Afin de configurer le Smart-hub pour interagir avec ces systèmes, nous voyons qu'il faut spécifier les plug-ins (OPC UA pour le connecteur, OPC UA Projection pour les règles de projection et OPC UA Transformation pour les règles d'unification) pour chaque système externe comme illustrés dans la table 4.2.

Donc, les systèmes identiques partagent les mêmes composants de communication, de domaine spécifique, de domaine indépendant, etc. De la même manière, nous pouvons avoir d'autres systèmes identiques sur le réseau comme les systèmes SIG1 et SIG2 qui partagent les

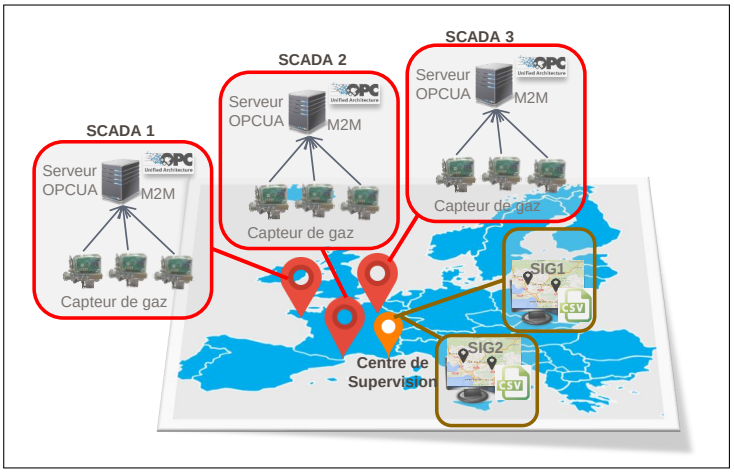


FIGURE 4.15 – Exemple des systèmes identiques dans le réseau de gaz

Role	System	Load plugins				Orchestration Mode
		Comunication	Domain Specific	Domain Independent	FetchAndProcess	
DP	SCADA1	OPCUA	OPCUAProjection	OPCUATransformation	-	Pull Périodique
DP	SCADA2	OPCUA	OPCUAProjection	OPCUATransformation	-	Pull Périodique
DP	SCADA3	OPCUA	OPCUAProjection	OPCUATransformation	-	Pull Périodique
..	...	..	...	...	...	...
DC	SIG1	FileSystem	CSVProjection	TabularTransformation	SIGDataProcessing	Périodique
DC	SIG2	FileSystem	CSVProjection	TabularTransformation	SIGDataProcessing	Push

TABLEAU 4.2 – Exemple de configuration spécifique au Smart-hub pour interagir avec des systèmes identiques

mêmes plug-ins. Afin de simplifier l'utilisation de l'architecture et d'empêcher l'utilisateur de répéter la même configuration pour des systèmes identiques, le Smart-hub distingue deux termes :

- Un **Type de système (System-Type)** : ce terme indique une abstraction du concept d'un système qui partage des composants similaires. Par exemple, nous donnons le nom "GasAnalyzerSystem" et "GeolocalizationSystem" pour indiquer l'abstraction de tous les systèmes identiques SCADA d'analyseurs de gaz et systèmes SIG respectivement.
- Un **Instance de système (System-Instance)** : ce terme indique un système concret qui utilise les composants identifiés par son type de système. Par exemple, les systèmes concrets (SCADA1, SCADA2, SCADA3, etc.) et (SIG1, SIG2, SIG3) utilisent les composants définis par son type de système GasAnalyzerSystem et GeolocalizationSystem respectivement.

La table 4.3 montre un exemple d'utilisation de ces termes pour définir la configuration de l'exemple introduit précédemment. Cette distinction de deux termes n'a pas d'impact sur la fonctionnalité du Smart-hub. Cependant, il s'agit d'un choix de développement pour simplifier la tâche à l'utilisateur pour configurer le Smart-hub.

Role	System Type	Load plugins				Orchestration Mode	System Instance
		Comunication	Domain Specific	Domain Independent	SOS Fetch AndProcess		
DP	GasAnalyzer System	OPCUA	OPCUA Projection	OPCUA Transformation	-	Pull Périodique	SCADA1, SCADA2, SCADA3, ...
DC	Geolocalization System	FileSystem	CSV Projection	Tabular Transformation	SIGData Processing	Push Périodique	SIG1, SIG2

TABLEAU 4.3 – Exemple de configuration avec les deux termes type de système et instance de système

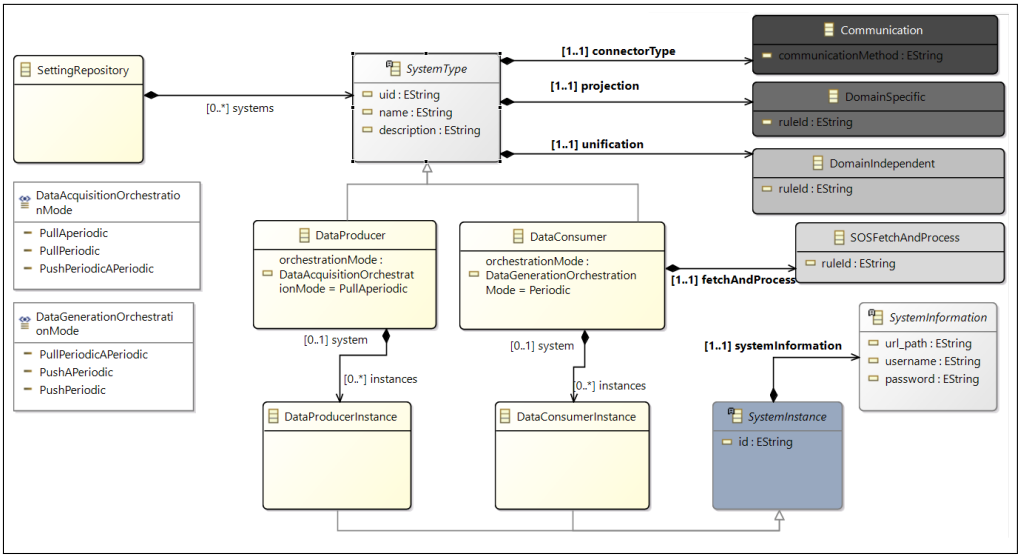


FIGURE 4.16 – le métamodèle de configuration "Smart-hub Dedicated Configuration MetaModel"

En matière de mise en œuvre, nous avons défini un métamodèle nommé "Smart-hub Dedicated Configuration MetaModel" comme montré dans la figure 4.16 pour définir ces configurations. Chaque type de système est associé avec ses propres plug-ins pour étendre le composant de communication, de domaine spécifique, de domaine indépendant. Un type de système peut être un type de producteur de données ou un type de consommateur de données. Il faut indiquer le mode d'orchestration de chacun. Cependant, un type de consommateur de données est associé aussi avec un plug-in pour étendre le sous-composant de traitement de données (SOSFetchAndProcess). Enfin, pour chaque type de producteur de données (ou chaque type de consommateur de données), nous pouvons ajouter la liste des instances de systèmes de producteur de données (ou des instances de consommateur de données). Chaque instance de système a des informations complémentaires qui sont accessibles via le concept "SystemInformation". Ces informations sont expliquées dans la section suivante. Ce méta-modèle est instancié pour définir la configuration de différents environnements. Figure 4.17 illustre un exemple de configuration (une instance de métamodèle SettingsRepository) qui définit trois instances de systèmes DP (SCADA1, SCADA2, SCADA3) et de deux instances de systèmes DC (SIG1, SIG2) partageant les mêmes composants définis

par leurs types de systèmes GasAnalyzerSystem et GeolocalizationSystem, respectivement et qui correspond à l'exemple de la table 4.3.

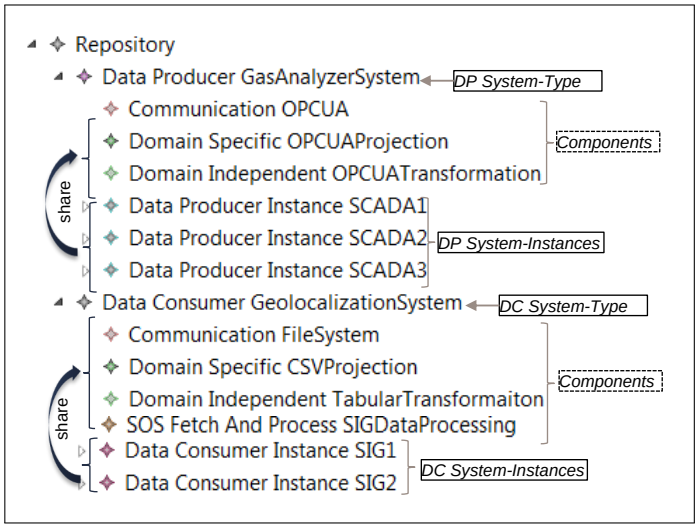


FIGURE 4.17 – Exemple de la configuration spécifique au Smart-hub

4.2.4.2 La configuration spécifique au système externe

Dans les paramètres de l'instance système, des informations supplémentaires, concernant une instance de système particulière telle que l'authentification, adresse IP, URL peuvent être définies. Dans le métamodèle de configuration spécifique au Smart-hub, nous avons défini le concept abstrait "SystemInformation". Ce concept est entendu avec un autre métamodèle "System Dedicated Configuration MetaModel" qui contient des concepts pour définir les informations complémentaires selon l'environnement d'exécution de Smart-hub (cf.4.18). Par exemple, nous étendons le métamodèle avec les concepts pour ajouter les paramètres d'un système avec un connecteur OPC UA , SOAP ou Rest, etc. ou pour ajouter les données de filtrage, etc. Grâce aux propriétés du modèle EMF "Child Creation Extenders" et "Extensible Provider Factory", l'édito de configuration de Smart-hub, sera capable de charger ces paramètres sans compiler le code de configuration spécifique au Smart-hub. Pour plus de détails sur ces propriétés veuillez-vous référer à [Steinberg et al., 2008]. Ce couplage des paramètres à deux niveaux rend le Smart-hub plus modulaire en raison de sa capacité à ajouter des informations d'instance système pour un environnement spécifique que nous ne connaissions pas auparavant et qui change en fonction de l'environnement, sans affecter le référentiel de configuration au niveau spécifique du Smart-hub.

La figure 4.19 montre un exemple de configuration spécifique des systèmes externes pour les systèmes basés OPC UA et SIG. Chaque instance de système a des informations différentes, par exemple SCADA1 utilise une adresse IP, un mot de passe et une fréquence qui sont différentes de SCADA2 et SCADA3.

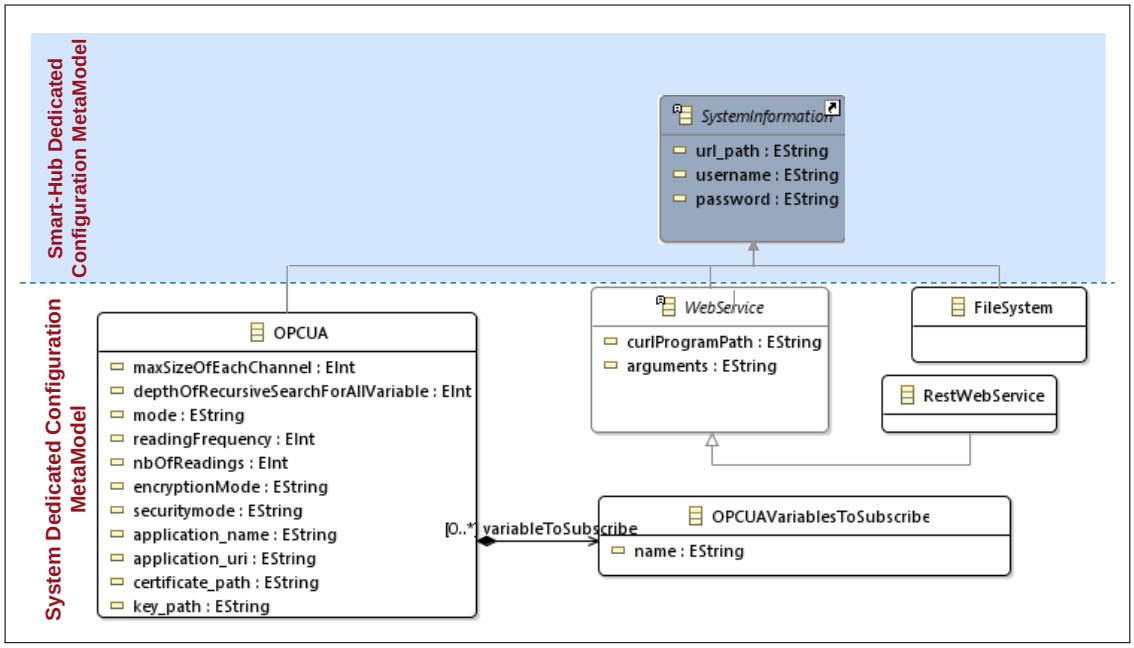


FIGURE 4.18 – Le métamodèle de configuration spécifique au système externe "System Dedicated Configuration MetaModel"

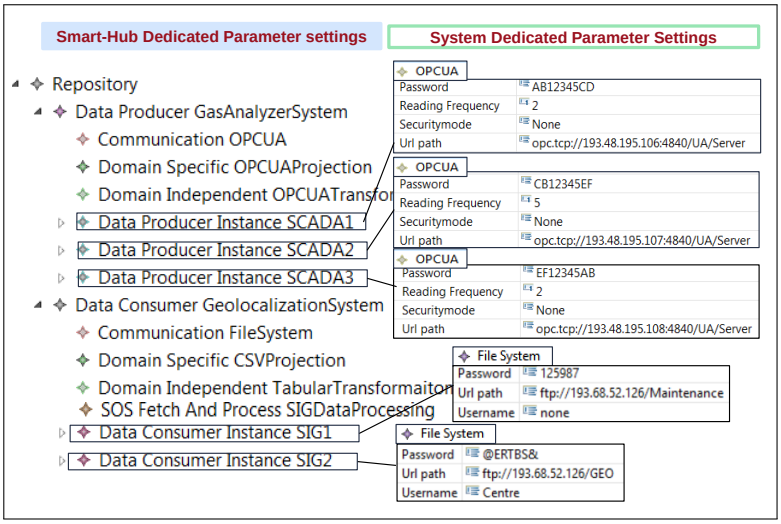


FIGURE 4.19 – Exemple de la configuration spécifique aux systèmes externes

4.3 Conclusion

Pour réaliser l'architecture Smart-hub décrite au chapitre 3, ce chapitre a présenté un prototype de mise en œuvre de l'approche en utilisant un certain nombre d'outils et de techniques de développement. Nous avons commencé en introduisant ces outils tels que la plates-forme de développement Eclipse, le concept de plug-ins, le cadre de modélisation EMF, le langage de transformation ATL, et les outils de projections. Ensuite, nous avons utilisé ces techniques

pour mettre en œuvre l'architecture Smart-hub. Nous décomposons l'implémentation en trois blocs : "bloc d'interopérabilité", "bloc d'orchestration" et "bloc de paramétrage".

Le bloc d'interopérabilité gère les problèmes de l'interopérabilité au niveau technique, syntaxique et sémantique en implémentant les quatre composants horizontaux de l'architecture. Pour chaque composant horizontal, nous avons défini les fonctionnalités fournies et ajouté le concept de plug-in afin de rendre le composant extensible. D'abord, nous avons implémenté le composant de communication qui gère l'interaction avec les systèmes externes via un catalogue de connecteurs qui sont ajoutés comme plug-ins. Ensuite, nous avons utilisé les techniques de projection et de modélisation dans le composant spécifique au domaine pour définir les métamodèles de chaque domaine/système et les règles de projections ajoutées comme plug-ins. Dans le composant indépendant du domaine, nous avons défini un metamodèle agnostique, générique et indépendant de domaine spécifique "CDM" pour unifier les sémantiques de données. Grâce aux outils de transformation de modèles comme ATL, nous pouvons ajouter des règles de mapping entre les métamodèles des systèmes et ce metamodèle CDM comme plug-ins. Enfin, au niveau de composant du système de systèmes, nous avons introduit comment nous agrégeons les données et les retirons au/de dépôt central. Nous avons implémenté une méthode générique pour agréger les données. Ensuite, en correspondance avec l'approche de traitement (post-agrégation), nous avons utilisé ATL pour charger et traiter les données qui sont ajoutées comme plug-in selon les besoins définis via les règles métiers/opérationnelles.

Le bloc d'orchestration gère l'orchestration de données entre les systèmes externes et entre les composants de Smart-hub. Nous avons défini les scénarios de flux de données et les différentes options d'interaction pour l'acquisition et la génération de données. Cependant, nous avons seulement implémenté les options Pull périodiques et nous avons adapté l'option Push Périodique en utilisant les concepts d'appel de méthodes. De plus, nous avons utilisé les techniques de programmation concurrente via l'api "Java Concurrency" pour exécuter les processus de DA et DG simultanément en temps réel.

Le bloc de paramétrage implique le composant de paramétrage du Smart-hub. Nous avons utilisé deux niveaux de métamodèles pour implémenter la configuration. Le premier niveau gère la configuration spécifique au Smart-hub alors que le niveau inférieur gère la configuration spécifique aux systèmes externes.

Dans le chapitre suivant, nous montrerons comment ces composants fonctionnent ensemble pour un certain nombre de cas d'utilisation.

Troisième partie

Cas d'études, Validation et Conclusion

Chapitre 5

Illustration de la solution développée

Sommaire

Dans ce chapitre du manuscrit, nous présentons quelques cas d'étude afin de démontrer l'application du Smart-hub dans certains environnements où l'interopérabilité est l'un des problèmes importants à gérer. La première section décrit l'application du Smart-hub sur l'exemple fil rouge présenté dans les chapitres précédents. La deuxième section présente l'application du Smart-hub sur le cas d'usage du projet GONTRAND afin de répondre à notre objectif F7. Ce deuxième exemple nous montre la façon dont le Smart-hub promeut la réutilisation des plug-ins et l'agilité de la solution pour une adaptation aux changements. Il montre également la capacité du Smart-hub à fonctionner comme un agrégateur de données provenant de nombreuses sources hétérogènes et à fournir les données pour un système d'hyperviseur.

5.1 Démonstration de l'exemple fil rouge via le Smart-hub

Cette section présente l'application du Smart-hub sur l'exemple fil rouge présenté dans la section 3.1. Nous avons déjà discuté des composants requis pour les fonctionnalités du Smart-hub. Nous reverrons ces composants et nous montrerons comment ils sont assemblés pour atteindre notre objectif.

5.1.1 Préparation de plug-ins

Tout d'abord, une liste des plug-ins et des metamodèles spécifiques au domaine est développée tel qu'illustré dans la figure 5.1. Ils servent à étendre des composants du Smart-hub afin d'interagir avec les systèmes externes (SCADA1, SCADA2, GMAO, SIG).

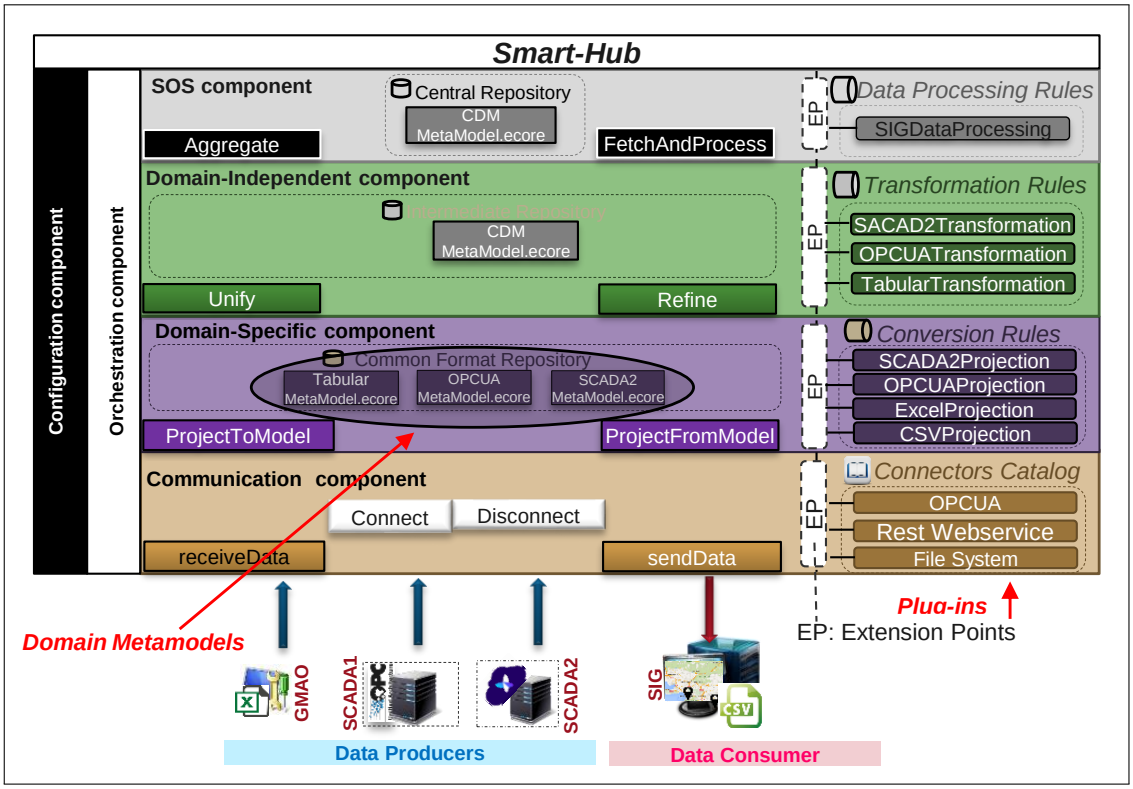


FIGURE 5.1 – Les plug-ins et les metamodèles pour l'extension du Smart-hub

Au niveau de composant de communication, les plug-ins sont :

- **OPC UA** pour la communication avec les serveurs OPC UA (ici SCADA1).
- **RestWebservice** pour la communication avec les systèmes conformes au service web Rest (ici SCADA2).
- **FileSystem** pour l'interaction avec les fichiers (utilisés par les systèmes GMAO et SIG).

Au niveau de composant spécifique au domaine, nous avons défini quelques métamodèles spécifiques au domaine : **OPC UA**, **Tabular**, **SCADA2**. Ces métamodèles sont utilisés par des plug-ins suivants :

- **OPCUAProjection** pour la projection de données OPC UA vers le modèle OPC UA (injecteur).
- **SCADA2Projection** pour la projection des données fournies par le système SCADA2 vers le modèle de SCADA2 (injecteur).
- **ExcelProjection** pour la projection de format Excel vers un modèle tabulaire (injecteur) ;
- **CSVProjection** pour la projection de modèle tabulaire vers le format CSV (extracteur).

De même au niveau de composant indépendant du domaine, nous avons réalisé les plug-ins suivants :

- **OPCUATransformation** pour transformer le modèle OPC UA en modèle unifié CDM ;
- **TabularTransformation** pour transformer le modèle tabulaire vers CDM et vice versa ;
- **SCADA2Transformation** pour transformer le modèle spécifique au domaine vers CDM.

De même, un plug-in est développé pour étendre la fonctionnalité FetchAndProcess du composant système de systèmes. Un composant **SIGDataProcessing** est développé pour traiter les données agrégées dans le référentiel central afin de générer les données requises par le système SIG.

5.1.2 Configuration du Smart-hub

Ensuite, le Smart-hub est configuré afin d'agréger les données des producteurs de données SCADA1, SCADA2, GMAO et de générer des données traitées pour le consommateur de données SIG. Un extrait de la configuration est montré dans la figure 5.2. La configuration implique deux niveaux, la configuration spécifique au Smart-hub pour charger des plug-ins pour chaque système (cf. la figure 5.2 à gauche) et la configuration spécifique aux systèmes externes pour gérer les paramètres des systèmes externes (cf. la figure 5.2 à droit). Les détails techniques de la configuration sont déjà abordés dans la section 4.2.4.

5.1.3 Fonctionnement du Smart-hub

Le Smart-hub gère deux processus, l'acquisition de données de DP et la génération de données pour les DC comme expliqué dans la section suivante.

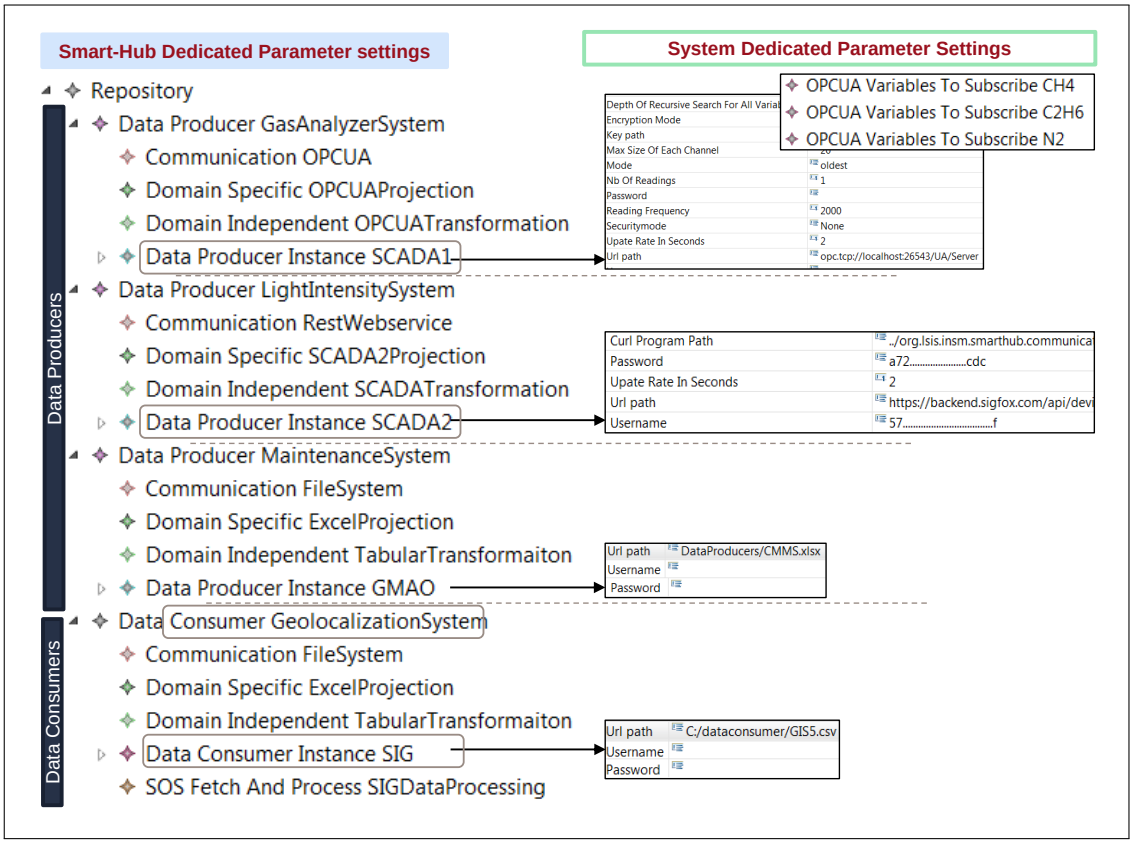


FIGURE 5.2 – La configuration du Smart-hub pour notre exemple fil rouge

Acquisition de données

Le processus d'acquisition de données (cf. figure 5.3 étapes 1-4) passe par un certain nombre d'étapes pour chaque système :

- Le Smart-hub fait une demande de pull pour le système SCADA1. Les données de SCADA1 sont acquises via le mécanisme de communication OPC UA. Le connecteur OPC UA est configuré pour s'abonner aux variables (CH4, C2H6, et N2). Ensuite, les données sont projetées vers un modèle OPC UA (DS_SCADA1.xmi) en utilisant le plug-in "OPCUAProjection" dans la couche spécifique au domaine. Après cela, le modèle est unifié en utilisant le plug-in "OPCUATransformation" indépendant du domaine (DI_-SCADA1.xmi). Enfin, ce dernier est agrégé dans le référentiel central.

- le Smart-hub lance une demande (pull) destinée au système SCADA2. Les données de SCADA2 sont acquises en utilisant le plug-in de service web. Puis les données sont projetées vers le modèle (DS_SCADA2.xmi) conformant au metamodèle SCADA2MetaModel en utilisant le composant spécifique au domaine "SCADA2Projection". Après cela, le modèle est unifié en utilisant le composant in-

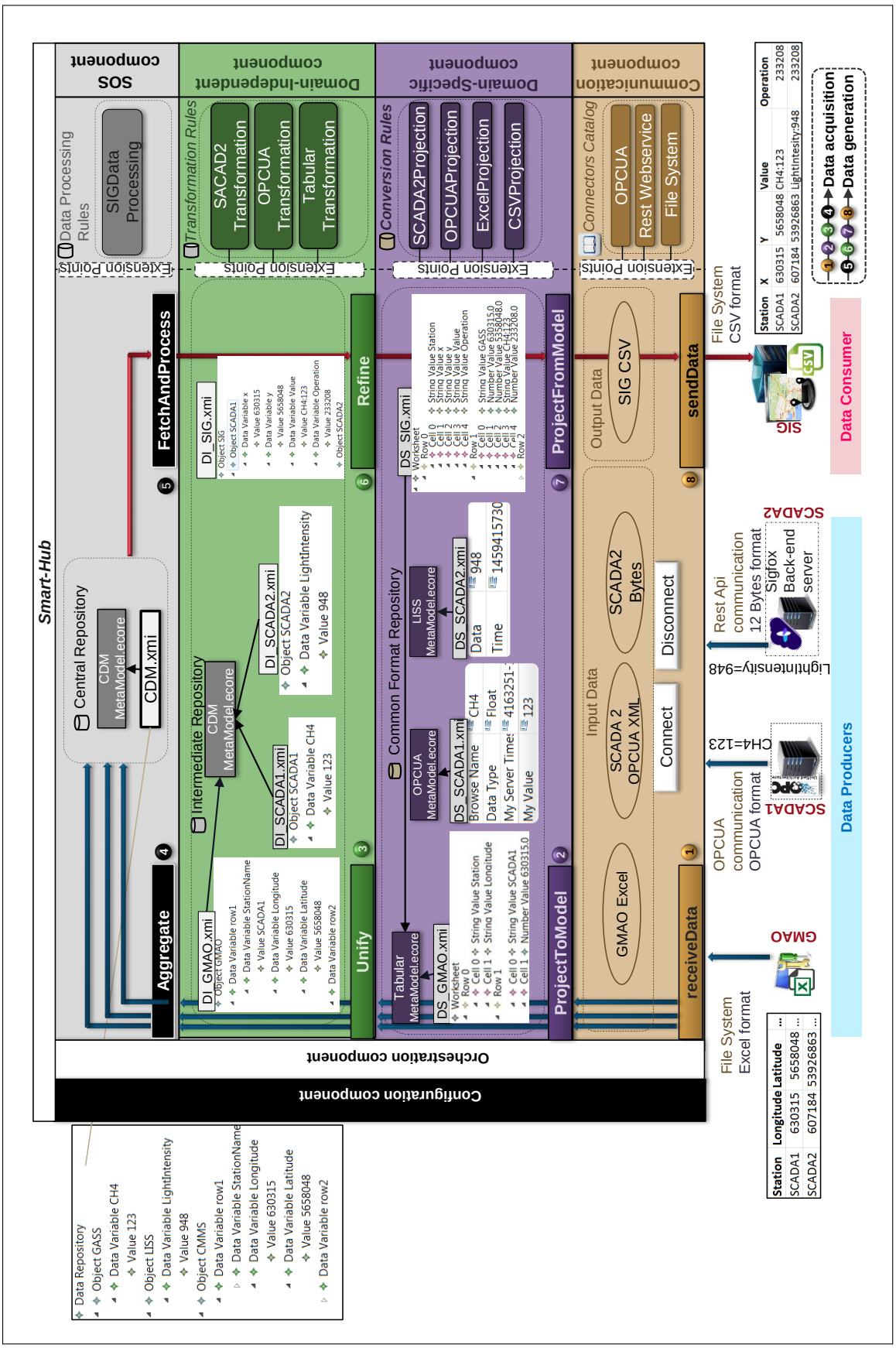


FIGURE 5.3 – Démonstration de l'exemple étude via le Smart-hub

dépendant du domaine "SCADA2Transformation" (DI_SCADA2.xmi), et finalement agrégé dans le référentiel central.

- le Smart-hub demande (pull) les données dans l'emplacement de fichier de GMAO. Ensuite, le fichier est chargé dans le Smart-hub via le connecteur de fichier "FileSystem". Puis, les données sont projetées vers le modèle (DS_GMAO.xmi) conformément au metamodèle "Tabular" en utilisant le composant spécifique au domaine "TabularProjection". Après, le modèle (DI_GMMAO.xmi) est unifié en utilisant le plug-in "TabularTransformation" et finalement agrégé dans le référentiel central.

Génération de données

Le processus de génération de données (cf. figure 5.3 étapes 5-8) met à disposition les données pour les consommateurs de données. Dans cet exemple, nous avons un seul consommateur de données pour qui les données doivent être fournies. Les données sont poussées périodiquement du CDM et mises à disposition pour le DC. D'abord, le plug-in de FetchAndProcess "GISDataProcessing" est appliqué sur le modèle CDM pour générer un modèle (DI_SIG.xmi) dans la couche indépendante du domaine. Ce modèle traite les données en appliquant les règles métiers/opérationnelles requises par le système SIG. Ensuite, ce modèle est affiné en utilisant le plug-in "TabularTransformation" dans le composant indépendant au domaine (DS_SIG.xmi). Après cela, les règles de composants spécifiques au domaine "CSVProjection" sont appliquées pour générer le fichier CSV qui est finalement à la disposition du répertoire de système SIG en utilisant le connecteur du système de fichiers.

5.1.4 le Smart-hub dans l'architecture dirigée par les modèles

La figure 5.4 montre le comportement du Smart-hub entre l'espace technique du système et l'espace de modélisation. Autrement dit, il montre comment les techniques de l'architecture dirigée par les modèles sont utilisées à l'intérieur de Smart-hub. Les étapes deux et sept représentent la projection et les étapes de trois à six représentent la transformation entre les modèles via le modèle pivot. Les étapes un et huit représentent la communication avec le système externe (elles n'ont aucun rôle dans la figure).

5.1.5 Synthèse

Dans cette section, nous avons illustré comment le Smart-hub peut gérer le problème d'interopérabilité entre les systèmes hétérogènes. Les systèmes hétérogènes ont pu échanger les données en temps réel sans imposer de normes/techniques entre les systèmes. Le Smart-hub gère également les règles métiers/opérationnelles afin de traiter les données avant de les mettre à la disposition pour le consommateur de données (le consommateur de données

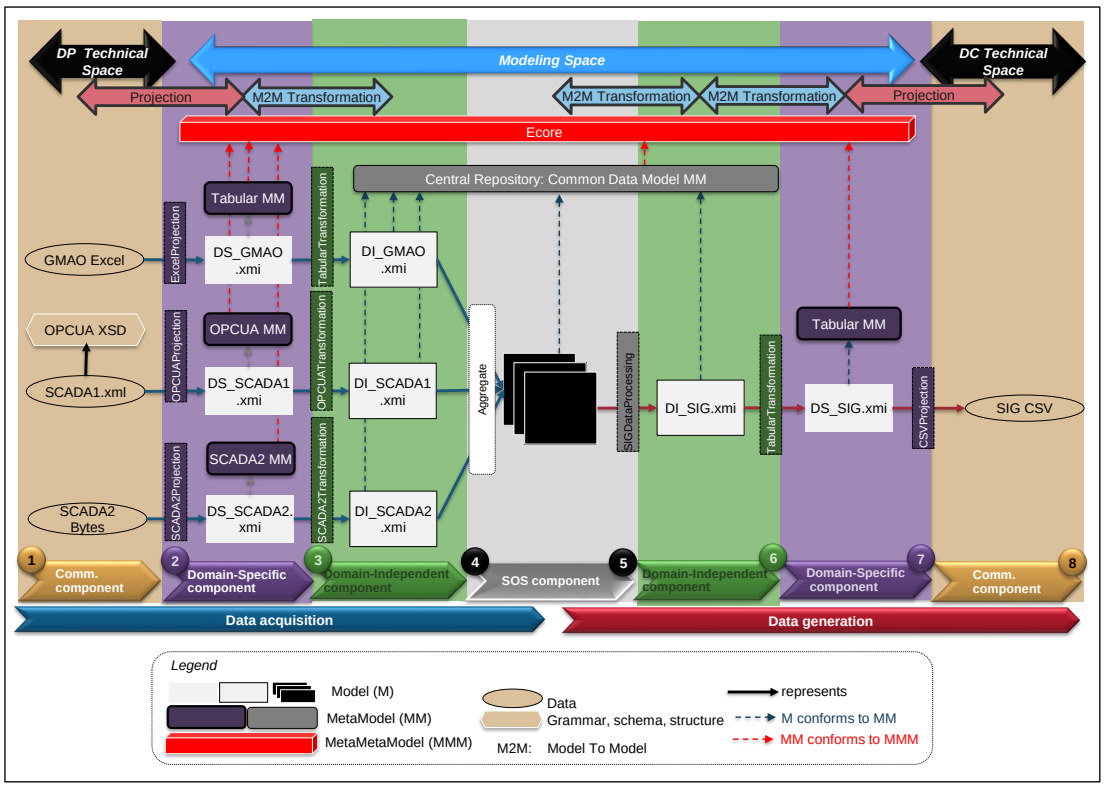


FIGURE 5.4 – Running Example in MDE

en a besoin dans une sémantique spécifique). La modularité a permis d'étendre le Smart-hub pour s'adapter à l'environnement actuel en termes de plug-ins qui sont développés une fois et peuvent être utilisés plusieurs fois comme cela sera illustré dans la section suivante.

5.2 Cas d'usage : Project GONTRAND

Dans le réseau de gaz actuel, il y a plusieurs distributeurs de gaz : GRDF, GDS et Régaz. Chaque distributeur assure la distribution du gaz naturel pour les millions de clients en France. L'acheminement du gaz naturel, depuis les centres de production, les points de stockages et les terminaux méthaniers, jusqu'aux consommateurs finaux, se fait sur de grandes distances par l'intermédiaire de méthaniers ou de gazoducs. Afin d'assurer une pression minimale de service aux consommateurs en bout de chaîne et de limiter l'agrégation d'impuretés, le gaz est maintenu à des pressions élevées dans ces gazoducs. Il arrive alors en périphérie des pôles de consommation et est injecté dans le réseau du distributeur de gaz. Ensuite, le distributeur gère l'abaissement de pression (via des postes de détente - PDR) afin de transporter le gaz (via des postes injection transport - PIT). Enfin les distributeurs abaissent la pression en centaines de millibars au niveau du consommateur final (via des postes de détente - PDR). Dans le cadre du projet comme GONTRAND [1], les distributeurs gèrent l'intégration de biométhane dans les réseaux de distribution (via des postes d'injections de biométhane - PBM). Pour conclure, chaque distributeur gère différents types de postes (poste de détente, poste injection transport et postes d'injections de biométhane) qui sont distribués sur plusieurs sites. Chaque site est surveillé via des systèmes OT (cf. 1.1) tels que des systèmes d'acquisitions SCADA. Chaque site peut également avoir ses propres systèmes d'information IT (cf. 1.1) tel que SIG :QGIS, GMAO, etc.

Le distributeur GRDF gère deux sites de distribution : un site avec un poste d'injection de biométhane appelé 'Aquapole' ; un deuxième site appelé "Reseau Fictif" gère sept postes divers entre des postes de détentes (appelée Jeanne d'Arc, Plobsheim, Gare, Lipsheim), des postes injection transports (appelée Garonne et Cor de chasse) et des postes d'injections de biométhane (appelée Biométhane). Les télémesures du premier site sont les données terrain actuellement remontées et sont accessibles via OPC UA. Cependant, les données du deuxième site ne sont pas accessibles pour s'intégrer directement en raison de certains aspects de confidentialité. Nous avons donc un simulateur de réseau de distribution de gaz qui simule l'acquisition de données pour ce site de distributeur GRDF. Ce simulateur met à disposition les télémesures via OPC UA.

Le distributeur REGAZ gère un site appelé "REGAZ" qui gère trois postes de détente (Crespy, Brun, Charcot). Ce site met à disposition les télémesures d'acquisition via des services web.

Le distributeur GDS gère le site qui nous appelons "GDS". Ce site gère quatre postes de détente (Brumath, Benfeld, Client GNV, Wantzenau) . Il met à disposition les télémesures d'acquisition via des fichiers Excel.

Il y a aussi le site central de supervision appelé SiteCockpit. Ce site est dédié à la visualisation, l'analyse et le pilotage du réseau en temps réel. Ce centre contient un cockpit de

[1] www.gontrand.net

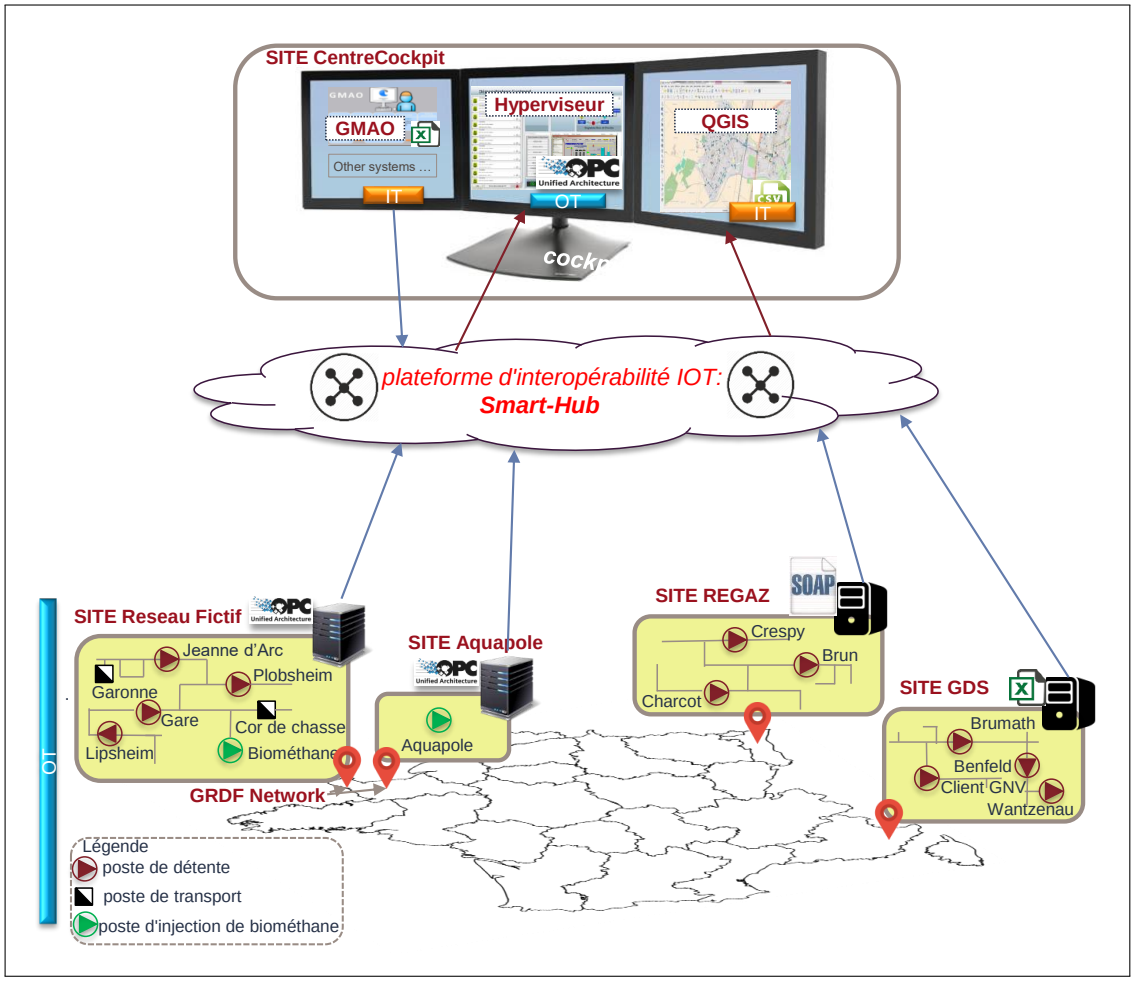


FIGURE 5.5 – Démonstrateur de projet GONTRAND

conduite qui assure un rôle d'hyperviseur qui vient en complément des SCADA existants et des systèmes d'information sur les sites différents (Aquapole, réseau fictif, REGAZ et GDS). L'IHM de cet hyperviseur utilise le protocole OPC UA afin de présenter l'état de télémesure de tous les systèmes sur tous les sites de tous les distributeurs. En plus, ce site inclut d'autres systèmes d'informations au niveau IT : le système GMAO qui fournit les informations de maintenance et de géolocalisation pour les systèmes d'acquisition distribués sur tous les sites. Ces informations sont fournies via des fichiers Excel ; le système SIG : QGIS qui visualise le réseau complet de gaz avec quelques informations en temps réel sur l'emplacement de chaque site sur la carte. Les données sont fournies par un fichier tabulaire partagé (format CSV).

Dans ce contexte, les systèmes de sites (Aquapole, réseau fictif, REGAZ et GDS) et le système (GMAO) au site central sont les producteurs de données (DP). Les systèmes hyperviseur et SIG :QGIS sont les consommateurs de données (DC). Les systèmes sont conformes ni à un standard commun ni à une technique commune à tous les niveau de l'interopérabilité.

5.2.1 L'intérêt de Smart-hub pour GONTRAND

Dans le réseau de gaz, chaque distributeur de gaz gère des systèmes hétérogènes. Ces systèmes fonctionnent de façon isolée, cependant il faut échanger les données avec d'autres systèmes afin de répondre aux objectifs globaux de l'entreprise. Ces systèmes hétérogènes sur les sites différents ne sont pas interopérables. Par exemple, le système hyperviseur est compatible seulement avec quelques sites basés OPC UA. Il faut également agréger les données de ces systèmes non interopérables avec OPC UA (Regaz et GDS). En plus, l'hyperviseur a été réalisé en tenant compte d'une politique de nommage prédéfinie par les règles métiers/opérationnelles du site central. Cette politique de nommage décrit comment les informations sont présentées par l'hyperviseur. Il faut donc transformer systématiquement les données fournies par les DP vers cette politique de nommage.

Nous présentons ici une version simplifiée de cette politique de nommage, cependant la politique implémentée dans le cadre de démonstrateur était plus complexe. Cette politique de nommage est composée d'un octet, suivi de deux tags de trois caractères suivi de l'information de télémesure comme suit : RESEAU-CATEGORIE-POSTE-INFORMATION (cf. figure 5.6). Le premier octet identifie le site : V pour le site de réseau fictif de GRDF ; G pour le site Aquapole de GRDF ; R pour le site de REGAZ ; S pour le site de distributeur GRDF. Le premier tag identifie le type de poste dans le site : PDR pour le poste de détente ; PIT pour le poste injection transport ; PBM pour postes d'injections de biométhane. Le deuxième tag identifie le poste via les premières trois lettres de son nom, par exemple GAR pour la poste GARE, BRU pour la poste Brumath. Le dernier tag identifie le nom de l'information, par exemple CH4, PressionAval, etc. Donc, il faut interoperer les systèmes de sites via le Smart-hub en prenants en compte les règles métiers/opérationnelles d'hyperviseur.

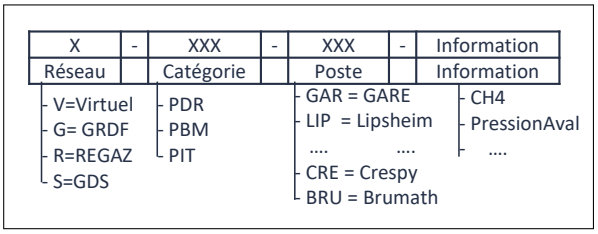


FIGURE 5.6 – Politique de nommage

De même, le système SIG :QGIS a besoin de données de localisation de chaque site. Ces données sont fournies par le système GMAO. Il a aussi besoin de quelques données d'acquisition calculées par le SCADA de chaque site. Les calculs sont définis par la règle business/opérationnelle qui doit trouver la moyenne de pression amonts, débit et CH4 sur chaque site où elle peut être appliquée (par exemple il n'existe pas de données de pression et débit sur le site Aquapole). Pour résumer, ces systèmes hétérogènes ne sont conformes ni à un standard commun ni à une technique commune. De plus, les consommateurs de données ont besoin d'avoir une sémantique prédéfinie afin de traiter les données.

La solution la plus simple serait de réaliser une solution ad-hoc spécifique à cet objectif. Selon le retour d'expérience des industriels, cette solution n'est pas : 1) modulaire parce qu'elle gère l'interopérabilité de façon holistique donc étroitement couplée; 2) extensible parce qu'elle est faite de systèmes spécifiques; 3) difficile à maintenir parce qu'elle n'est pas modulaire donc très coûteuse au niveau d'entretien; 4) agile, si des changements surviennent dans l'environnement, la solution ad-hoc doit être retravaillée.

Donc, le Smart-hub aide les développeurs à avoir une solution agnostique qui n'impose aucune standard ou technologie pour se conformer aux systèmes. Il peut être étendu pour s'adapter à la technologie/norme du système auquel il se connecte. La modularité à tous niveaux d'interopérabilité a pour effet de favoriser la réutilisation et de faciliter la maintenance. Cela rend également le système plus agile pour s'adapter aux changements en l'ajustant en fonction des changements dans l'environnement. Enfin, le Smart-hub prend en charge le traitement de données afin de répondre au besoin de règles métiers/opérationnelles définies par l'entreprise. Ces règles peuvent également être adaptées, réutilisées en fonction des besoins de l'environnement. Dans la section suivante, nous présenterons l'application du Smart-hub pour ce cas usage et nous finirons en synthétisant notre solution.

5.2.2 Préparation de plug-ins

Dans la partie suivante, nous présentons les plug-ins requis pour ce cas d'étude GONTRAND et introduisons leur application au sein du Smart-hub afin de promouvoir l'interopérabilité entre les systèmes hétérogènes.

La table 5.1 montre les plug-ins utilisés pour promouvoir l'interopérabilité entre les systèmes introduits précédemment. Quelques plug-ins existants ont été déjà utilisés dans notre exemple fil rouge, quelques-uns sont étendus et d'autres sont ajoutés. Des plug-ins existants tels que "TabularTransformaiton" et "OPCUATransofortmation" gèrent seulement le processus d'acquisition de données. Ils seront donc étendus pour gérer également le processus de génération de données (DG), comme cela sera expliqué dans la section suivante. D'autres plug-ins doivent être ajoutés afin de répondre aux besoins des systèmes dans le contexte de ce cas d'étude.

.

	Comunication	Domain Specific	Domain Independent	SOS FetchAndProcess
Existing	OPCUA	OPCUAProjection	OPCUATransformation*	
Existing	FileSystem	ExcelProjection	TabularTranformaiton*	
Existing	WebService Rest	CSVProjection		
New	WebService SOAP	REGAZProjection	REGAZTransformation	HyperviseurDataProcessing
New	OPCUAServer			QGISDataProcessing
				* Extended to handle data generation (DG)

TABLEAU 5.1 – La liste des plug-ins requis (existants, nouveaux et modifiés)

Les plug-ins de DP

Un certain nombre de plug-ins sont requis par les DP afin d'établir le processus d'acquisition de données. Les plug-ins existants "OPC UA" (client), "OPCUAProjection" et "OPCUATransformaiton" sont re-utilisés au niveau de composant de communication, spécifique au domaine et indépendante au domaine, respectivement par les systèmes de site Aquapole et de site Réseau fictif. Les plug-ins de communication en web service SOAP "WebService SOAP", de projection "REGAZProjection" et transformation "REGAZTransformation" de modèle de données ont été crées pour interagir avec le site Regaz. Les plug-ins existants de "FileSystem", "ExcelProjection" et "TabularTransformation" sont re-utilisés au niveau de composants de communication, spécifique au domaine et indépendante au domaine, respectivement par les systèmes de site GDS et le système GMAO de site central.

Les plug-ins de DC

Les systèmes DC (Hyperviseur et QGIS) sont installés sur le site central. En ce qui concerne le système Hypervieur, il utilise un client OPC UA pour acquérir les données. Donc, nous créons un plug-in de serveur "OPCUAServer" au niveau du composant de communication qui met à disposition les données agrégées pour l'hyperviseur IHM. Le modèle de données de OPC UA est générique, cependant les plug-ins existants de OPC UA gèrent seulement l'injection et l'unification pour le processus d'acquisition de données, donc ils sont entendus pour gérer aussi l'extraction et le raffinage pour le processus de génération de données. Pour retirer et traiter les données agrégées, nous ajoutons un plug-in HyperviseurDataProcessing qui prend en compte la politique de nommage de données présentée précédemment.

Le système QGIS utilise les plug-ins existants "FileSystem", "CSVProjection" et "TabularTransformaiton" pour mettre à disposition les données via le fichier CSV. Cependant, il faut ajouter un plug-in de traitement "QGISDataProcessing" qui retire les données de localisation de GMAO, applique les règles métiers/opérationnelles sur les données d'acquisition de plusieurs sites et les met à la disposition du composant indépendant du domaine.

5.2.3 Configuration du Smart-hub

Après les préparations de plug-ins, le Smart-hub doit être configuré. Cela impose la nécessité d'étendre le métamodèle "System Dedicated Configuration MetaModel" pour gérer les paramètres des systèmes externes comme illustrés dans la figure 5.7. Nous étendons le métamodèle avec les concepts pour ajouter les paramètres d'un système avec un connecteur OPC UA Server et services web SOAP. Ensuite le Smart-hub est configuré en instanciant le métamodèle "Smart-hub Dedicated Configuration MetaModel" pour définir chaque système (DP ou DC) et charger les plug-ins appropriés. Cela implique également l'instanciation du métamodèle "System Dedicated Configuration MetaModel" avec les paramètres des systèmes comme illustrés dans la figure 5.8.

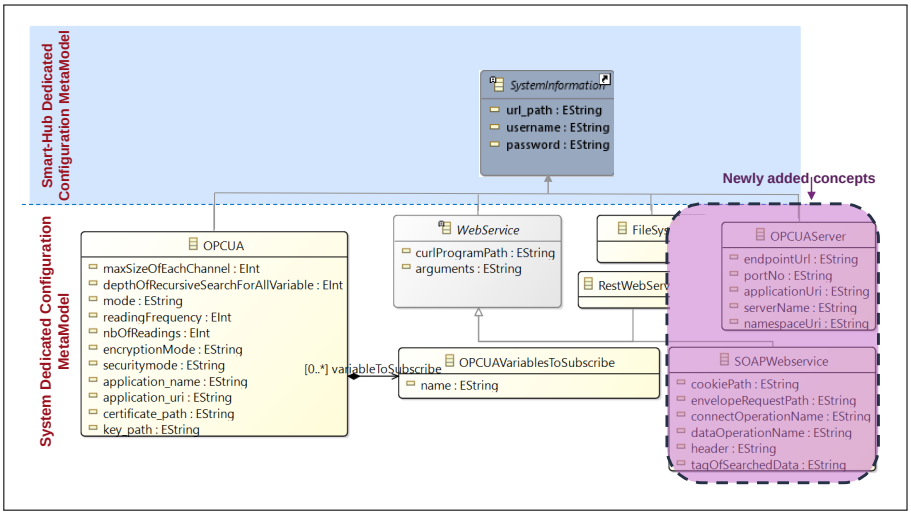


FIGURE 5.7 – Le métamodèle de configuration spécifique au système externe étendu "System Dedicated Configuration MetaModel"

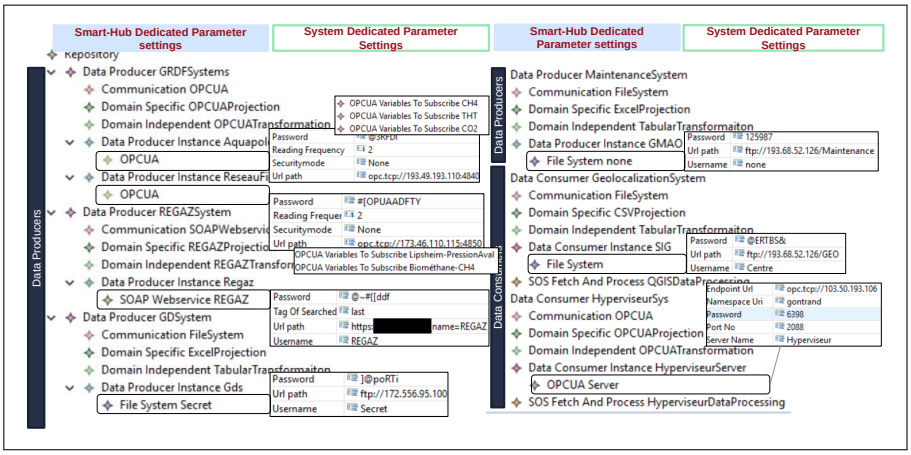


FIGURE 5.8 – La configuration du Smart-hub pour GONTRAND

5.2.4 Fonctionnement du Smart-hub

Les processus d'acquisition de données de DP et de génération de données pour des DC sont expliqués ci-dessous.

Acquisition de données

Le processus d'acquisition de données (cf. figure 5.9 étapes 1-4) passe par un certain nombre d'étapes pour chaque système :

- Le Smart-hub fait une demande de pull pour le système Aquapole (pareil pour le système de réseau fictif). Les données d'Aquapole sont acquises via le mécanisme de communication OPC UA. Ensuite, les données sont projetées vers un modèle OPC UA (DS_Aqupole.xmi et DS_RF.xmi) dans la couche spécifique au domaine en utilisant le

plug-in "OPCUAProjection". Après cela, le modèle est unifié en utilisant le plug-in "OPCUATransformation" indépendant du domaine (DI_Aqupole.xmi et DI_RF.xmi) et finalement agrégées dans le référentiel central.

- Le Smart-hub lance une demande (pull) destinée au système REGAZ. Les données de REGAZ sont acquises en utilisant le plug-in de communication service web SOAP, puis les données sont projetées vers le modèle (DS_REGAZ.xmi) conformément au metamodèle REGAZMetaModel en utilisant le composant spécifique au domaine "REGAZProjection". Après cela, le modèle est unifié en utilisant le composant indépendant du domaine "REGAZTransformation" (DI_REGAZ.xmi). Enfin les données de modèle sont agrégées dans le référentiel central.

- Le Smart-hub demande (pull) les données dans l'emplacement de fichier de site GDS. Ensuite, le fichier est chargé au Smart-hub via le connecteur de fichier "FileSystem". Puis, les données sont projetées vers le modèle (DS_GDS.xmi) conformément au metamodèle "Tabular" en utilisant le composant spécifique au domaine "TabularProjection". Après, le modèle (DI_GDS.xmi) est unifié en utilisant le plug-in "TabularTransformation" et finalement agrégé dans le référentiel central.

- Le Smart-hub lance une demande (pull) destinée aux emplacements de fichier de système GMAO. Ensuite, le fichier est chargé dans le Smart-hub via le connecteur de fichier "FileSystem". Puis, les données sont projetées vers le modèle (DS_GMAO.xmi) conformément au metamodèle "Tabular" en utilisant le composant spécifique au domaine "TabularProjection". Après, le modèle (DI_GMAO.xmi) est unifié en utilisant le plug-in "TabularTransformation" et finalement agrégé dans le référentiel central.

Génération de données

Le processus de génération de données (cf. figure 5.9 étapes 5-8) est expliqué ci-dessous :

- Le Smart-hub pousse périodiquement les données du CDM et les met à disposition pour l'Hyperviseur via le serveur DC OPC UA HyperviseurServer. D'abord, le plug-in de FetchAndProcess "HyperviseurDataProcessing" est appliqué sur le modèle CDM pour retirer et traiter les données en appliquant les politiques de nommage requis par l'Hyperviseur. En conséquence, un modèle (DI_HypSer.xmi) est généré dans la couche indépendante du domaine. Ensuite, ce modèle est affiné en utilisant le plug-in "OPCUATransformation" dans le composant indépendant au domaine (DS_HypSer.xmi). Après cela, les règles de composant spécifique au domaine "OPCUAProjection" sont appliquées pour générer le fichier de modèle de données (HypSer.xml) qui est finalement chargé par le connecteur ""OPCUAServer". L'IHM hyperviseur interagit avec ce serveur pour afficher les données.

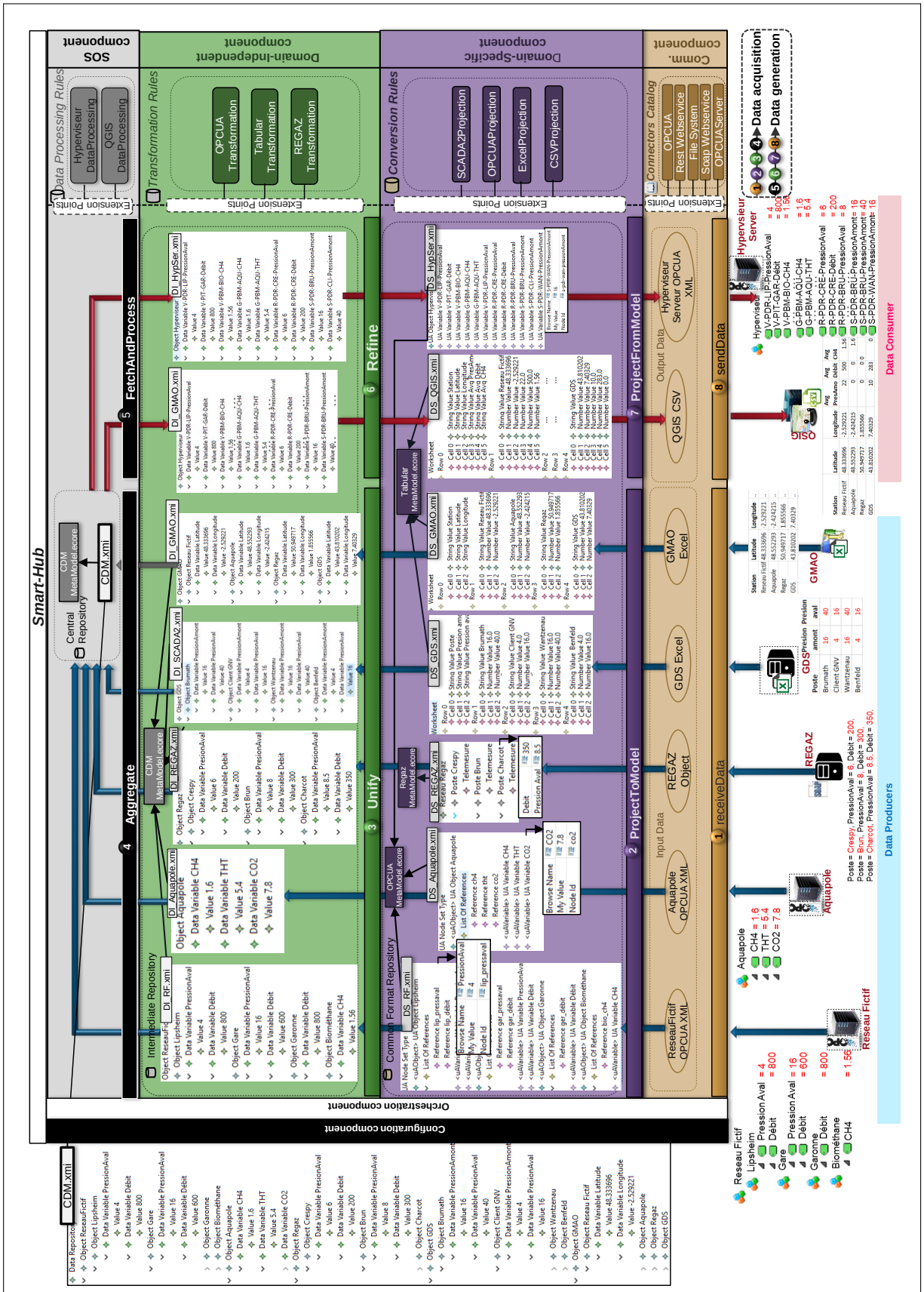


FIGURE 5.9 – Demonstration d'execution de Smart-hub pour GONTRAND

- Le Smart-hub pousse périodiquement les données du CDM et les met à disposition pour le DC QGIS. D'abord, le plug-in FetchAndProcess "QGISDataProcessing" est appliqué sur le modèle CDM pour retirer et traiter les données en appliquant les règles métiers/opérationnelles requises par le système QGIS. Donc, un modèle (DI_-QGIS.xmi) est généré pour la couche indépendante du domaine. Ensuite, ce modèle est affiné en utilisant le plug-in "OPCUATransformation" dans le composant indépendant au domaine (DS_QGIS.xmi). Après cela, les règles de composants spécifiques au domaine "OPCUAProjection" sont appliquées pour générer le fichier de modèle de données (HypSer.xml) qui est finalement chargé par le connecteur "OPCUAServer". L'IHM hyperviseur interagit avec ce serveur pour afficher les données.

5.2.5 Cas de usage étendu

Nous allons étendre notre cas d'usage en ajoutant et en modifiant certains systèmes externes comme représenté sur la figure 5.10. Le distributeur REGAZ a étendu son rôle dans la distribution de gaz. Par conséquent, il a mis à jour son site actuel REGAZ pour passer d'un système basé sur la communication SOAP vers un système basé sur MQTT. En plus, un autre site "REGAZ2" est installé qui gère deux postes d'injection de biométhane. Il met à disposition les données en OPC UA. Ce site a aussi une salle de supervision intermédiaire pour surveiller ce site en plus de site REGAZ. L'application IHM de supervision utilise le protocole OPC UA afin de présenter l'état des télémesures agrégées de ces deux sites. Enfin, le site de distributeur GDS a changé son format de données d'Excel à CSV.

Le système de producteurs de données de Site REGAZ est passé de SOAP à MQTT au niveau de communication. Par conséquent, il est nécessaire de créer un nouveau plug-in MQTT pour le composant de communication. Ensuite, le métamodèle "System Dedicated Configuration MetaModel" est entendu pour définir les paramètres de systèmes externes utilisant MQTT. Le Smart-hub est configuré pour charger ce nouveau plug-in pour ce système ainsi qu'avec les paramètres de ce système externe. Les plug-ins pour les autres composants ne sont pas affectés, ce qui est le résultat de la modularité de notre système. Le producteur de données de site REGAZ2 est basé sur OPC UA, donc il suffit de réutiliser les plug-ins préexistants de OPC UA au niveau de composants communication, spécifique au domaine et indépendant de domaine. Ensuite, il faut configurer le Smart-hub pour interagir avec ce système. Le site REGAZ2 a aussi un consommateur de données qui utilise un client OPC UA pour acquérir les données. Donc, il faut avoir un plug-in de serveur OPC UA au niveau de composant de communication. Comme ce plug-in existe déjà, il est suffit de le réutiliser. Les plug-ins au niveau de composant spécifique et indépendant au domaine sont génériques et ils peuvent être réutilisés. Cependant, il faut ajouter un nouveau plug-in de traitement de données qui charge seulement les données des sites REGAZ et REGAZ2. Nous n'avons aucun traitement spécifique à faire. Pour résumer, il suffit de créer un plug-in pour le traitement des données

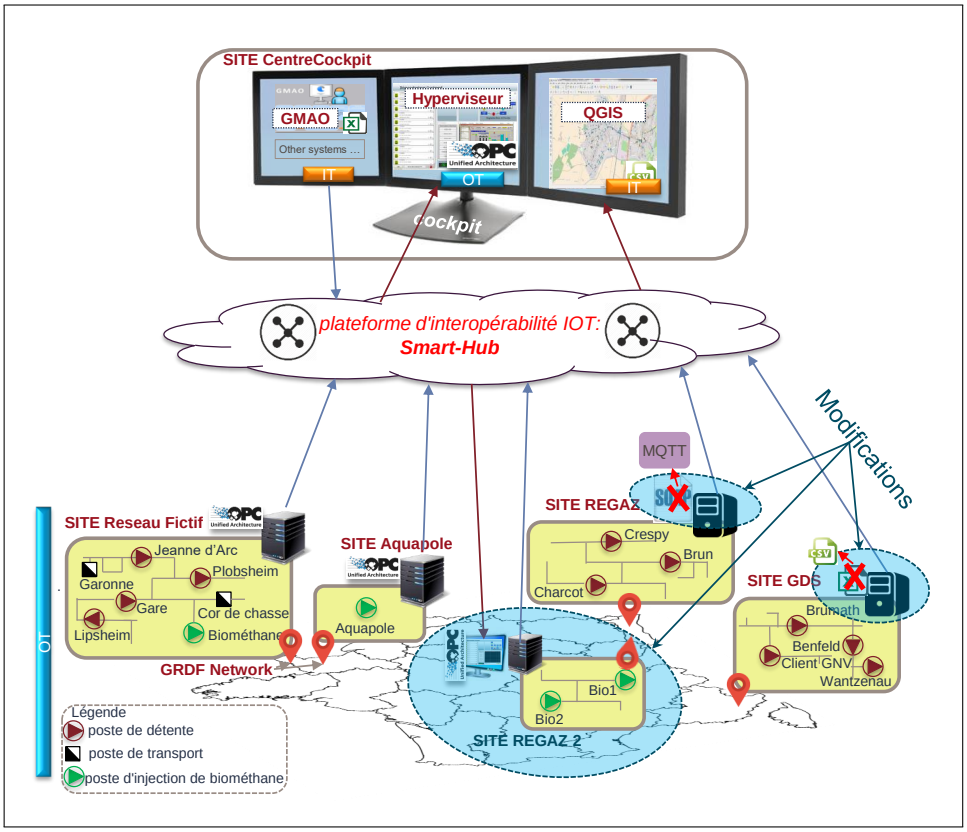


FIGURE 5.10 – Exemple entendu du démonstrateur de projet GONTRAND

afin de charger les données requises et de réutiliser les plug-ins préexistants. Le Smart-hub est configuré pour charger ces plug-ins de ce site et ensuite il est mis en exécution.

Le site de distributeur GDS a changé le format de données d'Excel à CSV. Comme les plug-ins de composant de communication et de composant indépendant de domaine sont modulaires, ils ne sont pas affectés par ce changement. Il est juste nécessaire d'utiliser le plug-in "CSVProjection" au lieu de "ExcelProjection" au niveau de composant spécifique au domaine. Le plug-in existant "CSVProjection" gère seulement l'extraction pour le processus de génération de données (utilisé par le système QGIS), donc il est entendu pour gérer aussi l'injection pour le processus d'acquisition de données. Ensuite, le Smart Hub est reconfiguré pour prendre en compte cette nouvelle configuration.

Ce cas d'usage a souligné les principales caractéristiques du Smart-hub :

- Générique : le Smart-hub n'impose pas un modèle de données ou une norme spécifique à respecter. Donc, il peut être appliqué dans n'importe quel domaine où l'interopérabilité doit être traitée. Ici, nous avons appliqué dans le domaine des réseaux de gaz.

- Modulaire : la modularité du Smart-hub en termes de différents niveaux d'interopérabilité a joué un rôle important dans notre proposition. Nous avons pu modi-

fier/remplacer le plug-in à un niveau d'interopérabilité sans affecter le plug-in pour les autres composants. Par exemple, dans le cas d' usage étendu de projet GONTRAND, nous avons remplacé les plug-ins à un niveau d'interopérabilité sans affecter les autres niveaux.

- Extensible : le Smart-hub est extensible à tous les niveaux d'interopérabilité et pour chaque niveau, nous sommes en mesure d'étendre les composants avec les plug-ins appropriés qui sont requis par les systèmes dans l'environnement.
- Agnostique : le Smart-hub est agnostique de la technologie/normes des systèmes avec lesquels elle interagit. Avec les extensions (les plug-ins), le Smart-hub a pu interagir avec les systèmes tels qu'ils existent sans leur imposer de suivre une technologie ou une norme spécifique.

La modularité et l'extensibilité ont eu un impact important sur la réutilisation et l'agilité du système. Un plug-in est créé une fois et utilisé plusieurs fois. De plus, lorsque l'environnement évolue, que de nouveaux systèmes sont ajoutés ou que des systèmes sont supprimés ou modifiés à un niveau d'interopérabilité particulier, le développeur peut rapidement réagir à ces changements en ajoutant les plug-ins requis et en configurant le Smart-hub. Il n'est pas nécessaire de retravailler le système d'interopérabilité de zéro. Cela se traduit par un gain global en termes de réduction du temps et du coût de développement, comme cela sera expliqué dans le chapitre suivant.

5.3 Conclusion

Ce chapitre a discuté l'application du Smart-hub à deux cas d'usage afin de démontrer comment utiliser notre système. Dans le premier cas d'usage de l'exemple fil rouge, nous avons créé un certain nombre de plug-ins afin d'étendre le Smart-hub pour interagir avec les systèmes externes de l'exemple. Nous avons montré comment configurer le Smart-hub pour charger ces plug-ins et comment il interagit avec les systèmes externes. Nous avons également montré comment le Smart-hub gère les données dans les deux processus pour l'acquisition de données et la génération de données des et aux systèmes externes, respectivement. Enfin, cet exemple s'est terminé avec la représentation du comportement du système vis-à-vis de l'architecture dirigée par les modèles.

La deuxième partie de ce chapitre visait à démontrer l'application de notre système sur le cas d'usage de projet GONTRAND. Ce cas d'usage a mis en évidence les principales caractéristiques de notre système telles que la modularité, l'extensibilité, l'agnositicité et l'aspect générique qui à leur tour favorisent la réutilisation et réduisent ainsi le coût de temps et du développement comme nous le verrons dans le chapitre suivant.

Chapitre 6

Validation

Sommaire

La validation est une phase essentielle de la théorie du design pour guider le développement et l'évaluation de nouvelles méthodes [Barth et al., 2011b]. L'IEEE définit la validation [IEE, 1998] comme "la confirmation, par la fourniture de preuves objectives, que les exigences spécifiées ont été remplies". Autrement dit, la validation détermine si la méthode où le logiciel final accomplit son but prévu en répondant aux exigences et aux attentes des utilisateurs. Maintenant que nous avons présenté notre solution, nous souhaitons fournir quelques arguments qui valident que celle-ci répond bien au besoin d'interopérabilité entre les systèmes hétérogènes et que celle-ci a aussi des valeurs ajoutées par rapport aux solutions existantes en validant les caractéristiques présentées précédemment : générique, modulaire, agnostique et extensibles.

Différentes approches de validation peuvent être utilisées pour valider les nouvelles solutions dans une grande variété de disciplines. Dans les disciplines de problèmes fermés qui utilisent des équations et des algorithmes mathématiques, la validation est un processus strictement formel, algorithmique, réductionniste, de confrontation, où les nouvelles connaissances sont vraies ou fausses. Cependant, dans certaines disciplines de problèmes ouverts, où les nouvelles connaissances sont associées à des heuristiques et à des représentations non précises, ainsi qu'à la modélisation mathématique, la validation n'est pas directement formelle. En fait, **la validation est considérée comme un processus graduel de renforcement de la confiance dans l'utilité de la nouvelle connaissance par rapport à un but** [Seepersad et al., 2006].

Il y a plusieurs façons et des études différentes proposant des cadres de validation tels que [Barth et al., 2011a, Seepersad et al., 2006]. [Frey and Dym, 2006] fournit une revue de la littérature de certaines approches de validation. Le cadre du carré de validation (Validation Square, en anglais) [Seepersad et al., 2006] est utilisé pour valider la recherche en conception et les méthodes de conception dans le domaine de l'ingénierie. Il convient de valider les recherches et les résultats scientifiques en général pour autant qu'ils puissent être appliqués à l'évaluation quantitative et qualitative. Dans ce manuscrit, nous utilisons ce cadre du carré de validation pour évaluer le Smart-hub comme expliqué dans la section suivante.

6.1 Le carré de validation dans le contexte du Smart-hub

Le carré de validation a pour but d'établir un processus de confiance pour valider les résultats scientifiques dans son utilité pour un but donné. Ici, notre but est d'assurer l'interopérabilité entre des systèmes hétérogènes en temps réel de manière à ce que chaque système puisse interagir en utilisant sa propre technologie (mécanisme de communication, son format de données et sa sémantique) en mettant l'accent sur les caractéristiques du systèmes générique, modulaire, agnostique et extensibles. Pour atteindre le but indiqué, nous avons défini quelques fonctions comme indiqué dans la section 1.5 du chapitre 1.

Dans ce cadre, l'utilité d'un résultat est associée à deux critères. Le premier critère est l'efficacité (effectiveness, en anglais) : est-ce que les résultats sont corrects ? Cela correspond à l'évaluation sur la base de mesures qualitatives. Le deuxième critère est l'efficience (efficiency, en anglais) : est-ce que les résultats sont obtenus avec des performances acceptables ? Cela correspond à l'évaluation sur la base de mesures quantitatives (ex. moins de temps et moins de dépenses) ? La figure 6.1 illustre le processus de renforcement de la confiance dans l'utilité où le carré de validation au bas du diagramme représente la synthèse du processus de validation.

Pour valider les résultats scientifiques, le carré de validation préconise alors de diviser l'exercice en quatre dimensions : (1) la validation théorique de la structure ; (2) la validation théorique de la performance ; (3) la validation empirique de la structure ; (4) la validation empirique de la performance ; dans la suite, nous appliquons chaque dimension dans le contexte du système Smart-hub.

6.1.1 Validation théorique de la structure

L'objectif principal de la validité structurelle théorique est la cohérence logique du système (notre résultat scientifique) proposé. La validation théorique de la structure impose trois étapes. La première étape consiste à définir les exigences de haut niveau qui doivent être satisfaites par notre système. Nous revoyons ces exigences qui ont déjà été définies dans la section 1.5.

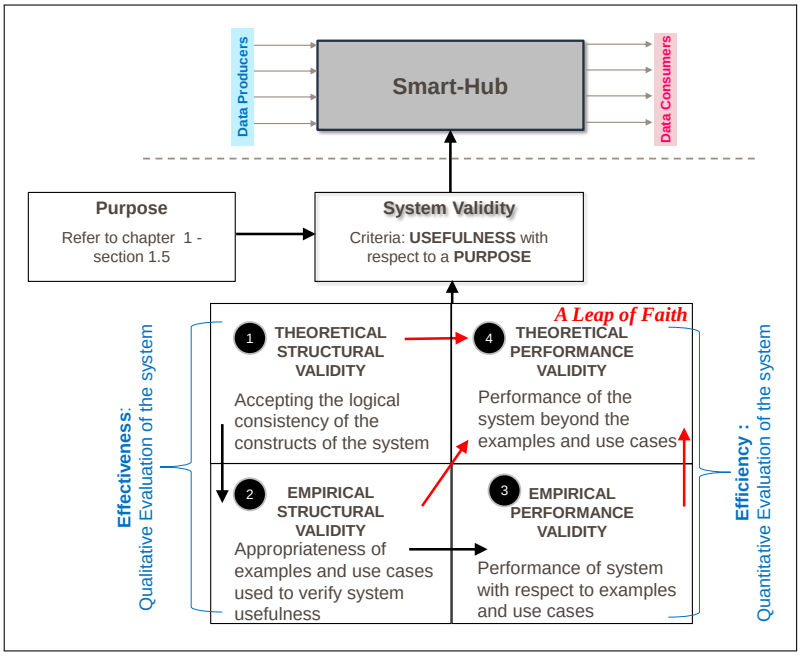


FIGURE 6.1 – Le processus de renforcement de la confiance dans l'utilité du Smart-hub [Seepersad et al., 2006]

- Exigence 1 : le système doit gérer plusieurs mécanismes de communication de façon agnostique et extensible.
- Exigence 2 : le système doit gérer et intégrer diverses syntaxes de façon agnostique et extensible.
- Exigence 3 : le système doit interpréter et intégrer diverses sémantiques de façon agnostique et extensible en appliquant les règles métiers/opérationnelles (connaissance) sur les données.
- Exigence 4 : le système doit acheminer (agréger et rendre disponible) les données en temps réel selon les options d'interaction pour des systèmes.
- Exigence 5 : le système conçu doit être générique, modulaire pour assurer l'interopérabilité entre systèmes hétérogènes.
- Exigence 6 : le système doit être réalisable techniquement.
- Exigence 7 : le système doit être testé sur un cas d'usage réel.

Ces exigences fournissent les mesures utilisées pour évaluer l'utilité du système pendant le processus de validation. La deuxième étape consiste à faire un état de l'art des solutions préexistantes vis-à-vis des exigences. Cet état de l'art a été réalisé dans le chapitre 1.8 qui a été divisé en trois sections où chaque section s'est terminée par une discussion sur des solutions existantes.

La dernière étape de la validité structurelle théorique est de valider la cohérence logique de notre système. Il peut être réalisé par différentes techniques logiques telles que des arguments logiques, des preuves mathématiques formelles ou informelles et les diagrammes de flux (flowchart) ou par différentes techniques empiriques telles que des exemples de problèmes conçus pour tester une capacité spécifique d'un système.

La figure 6.2 illustre un diagramme de flux pour montrer la collaboration et coopération entre tous les composants du Smart-hub pour aborder nos exigences. Après la configuration de Smart-hub, le Smart-hub charge les plug-ins afin de gérer des différents mécanismes de communication, diverses syntaxes, sémantiques et règles métiers/opérationnelles de chaque système. Ensuite, en fonction du rôle du système (DP or DC) le Smart-hub achemine (agréger et rendre disponible) les données. Si le système est un DP, le Smart-hub : 1) lit les données avec le mécanisme de communication de DP ; 2) projette les données vers un modèle spécifique au domaine en utilisant la règle de conversion de syntaxe associée avec ce DP ; 3) unifie la sémantique vers le modèle pivot en utilisant la règle de transformation de ce DP ; 4) agrège les données dans le référentiel central. Dans l'autre sens, si le système est un DC, le Smart-hub : 1) retire, traite les données du référentiel central et crée un modèle indépendant de domaine en utilisant les règles métiers/opérationnelles de ce DC ; 2) raffine le modèle indépendant de domaine vers un modèle spécifique au domaine en utilisant la règle de transformation de ce DC ; 3) projette le modèle spécifique au domaine vers des données en utilisant la règle de conversion de syntaxe associée à ce DP ; 4) alimente le DC avec les données et les rend disponibles en utilisant son mécanisme de communication. La collaboration et coopération entre tous les composants pour aborder nos exigences conduisent à une méthode complète pour répondre à l'interopérabilité. Nous pouvons dire que la méthode résulte d'un raisonnement qui lui assure une certaine cohérence logique. Donc, nous pouvons donc affirmer que la structure théorique de notre méthode est validée.

6.1.2 Validation empirique de la structure

L'objectif principal de la validation empirique de la structure est d'établir une confiance vis-à-vis des exemples (des cas d'études) dans un domaine spécifique. Nous avons choisi le domaine de la supervision de réseau de gaz pour illustrer et vérifier les performances de notre système. Dans cette phase de validation, les cas d'études sont décomposés, documentés et comparés en ce qui concerne nos exigences.

Dans ce manuscrit, nous avons deux cas d'usage principaux. Le premier cas d'usage est l'exemple fil rouge (cf. section 3.1 et 5.1) et le deuxième est le cas d'usage de projet GONTRAND (cf. chapitre 5). Pour le bien de cette discussion, nous avons décomposé les cas d'étude en plusieurs sous exemples :

- un exemple d'interaction du Smart-hub avec le producteur de données SCADA1.
- un exemple d'interaction du Smart-hub avec le consommateur de données SIG1.

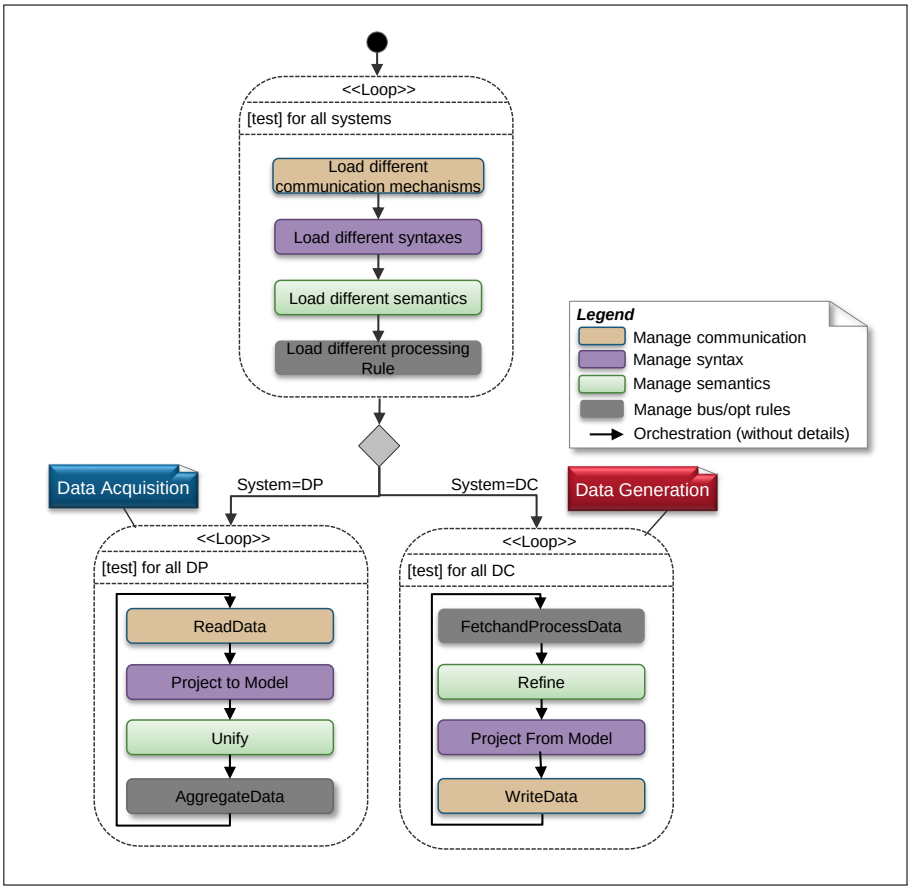


FIGURE 6.2 – Diagramme de flux du Smart-hub

- l'exemple fil rouge complète pour interagir avec tous les DP et DC. Il comprend les deux exemples précédents (SCADA1 et SIG1) .

- l'exemple du projet GONTRAND complet.

Le tableau 6.1 comprend la liste des exigences et des sous-fonctions auxquelles ces exemples sont destinés. Les premiers et deuxièmes sous-exemples ont été utilisés comme principales entités d'illustration dans les chapitres 3 et 4 pour traiter différents niveaux du problème d'interopérabilité. Ces deux exemples montrent comment le Smart-hub interagit avec les systèmes pour communiquer (lecture/écriture) des données, gérer et convertir les syntaxes, convertir et gérer la sémantique avec des règles métiers/opérationnelles. Le troisième exemple, qui fournit un environnement complet, montre comment le Smart-hub gère l'ensemble de composants, orchestre les données entre les composants et les systèmes externes, et met en évidence les caractéristiques : générique (indépendant de domaine), modulaire, extensible et agnostique. Le dernier exemple du projet GONTRAND renforce ces caractéristiques ainsi que l'agilité et la réutilisation qui sont les résultats de la modularité et l'extensibilité du système.

	Sous-exemple d'interaction avec DP (SCADA1)	Exemple d'Interaction avec DC (SIG)	Exemple fil rouge complète	Projet GONTRAND
Gérer des Mécanismes de communication				
Lire les données de DP	✓		✓	✓
Mettre à disposition les données pour DC		✓	✓	✓
Extensible			✓	✓
Agnostique			✓	✓
Gérer des diverses syntaxes				
Conversion au format commun	✓		✓	✓
La conversion de format commun		✓	✓	✓
Extensible			✓	✓
Agnostique			✓	✓
Gérer des diverses sémantiques				
Conversion à une sémantique commune	✓		✓	✓
Conversion d'une sémantique commune		✓	✓	✓
Application des connaissances métiers		✓	✓	✓
Extensible			✓	✓
Agnostique			✓	✓
Acheminer des données				
Acquisition de données	✓		✓	✓
Génération de données		✓	✓	✓
Orchestration			✓	✓
Architecture				
Générique (Indépendant de domaine)			✓	✓
Modulaire			✓	✓
Extensible			✓	✓
Agnostique			✓	✓
Agile				✓

TABLEAU 6.1 – Liste des fonctions du Smart-hub

La confiance que l'on peut avoir pour les résultats obtenus est étroitement liée à la qualité des exemples utilisés. Plus les exemples se rapprochent des cas que l'on rencontre dans un contexte opérationnel, meilleure est la confiance. Notre exemple est basé sur un cas d'usage existant qui est issu de l'industrie. Nous avons appliqué le Smart-hub sur des exemples qui interagissent avec des systèmes physiques réels en plus de certains simulateurs (où le système physique n'est pas accessible).

L'une des hypothèses simplificatrices est que nous avons utilisé quelques (deux ou trois) instances de systèmes pour chaque type de système. La seconde hypothèse simplificatrice est que nous travaillons avec les options d'interaction Pull et Push périodique pour le DP et DC respectivement. Cependant, cela n'a aucun effet sur le rôle de notre architecture dans le traitement de la question d'interopérabilité. Enfin, les tests de ces exemples évaluent l'applicabilité du Smart-hub, par conséquent, nous pouvons conclure que sa structure est empiriquement vérifiée grâce aux résultats obtenus à partir d'expériences menées sur des données représentatives des pratiques industrielles.

6.1.3 Validation empirique de la performance

La validation empirique de la performance implique de renforcer la confiance dans l'utilité d'un résultat scientifique en utilisant des exemples et des cases d'études. C'est une validation de l'utilité quantifiable. L'une des métriques de l'utilité est généralement liée à la réduction des coûts et/ou du temps et/ou à l'amélioration de la qualité. Le Smart-hub a été testé par des ingénieurs sur un certain nombre d'études de cas réels dans le réseau de distribution de gaz pour le projet GONTRAND. La modularité de l'architecture a conduit à concentrer et à développer un certain nombre de composants faiblement couplés qui s'attaquent aux différentes couches d'interopérabilité. Cela a abouti au développement d'un certain nombre de composants dans lesquels des plug-ins pour gérer différents mécanismes de communication, diverses syntaxes et sémantiques en prenants en compte les différentes règles métiers/opérationnelles sont développés pour tester la fonctionnalité du Smart-hub. Ensuite, les études de cas du projet GONTRAND ont été soumises à plusieurs itérations pour des changements de systèmes externes afin de tester la fonctionnalité de réutilisation et d'extensibilité, qui a été réalisée avec succès. Au niveau de l'amélioration de la qualité, les auteurs ont créé une comparaison des approches de l'interopérabilité dans l'état de l'art (cf. section 2.2.6). Nous avons fini par le fait qu'il existe des solutions génériques qui peuvent être appliquées à n'importe quel domaine, certaines solutions sont modulaires en termes d'interopérabilité, mais il n'y a pas de solution extensible à tous les niveaux d'interopérabilité et toutes les solutions imposent un standard/technique à respecter pour promouvoir l'interopérabilité. En outre, la plupart des travaux existants traitent l'interopérabilité soit entre les systèmes dans la portée IT (Information Technology) soit entre les systèmes dans la portée OT (Operational Technology), mais pas entre les systèmes dans les deux portées.

Cependant, le Smart-hub a été développé en tenant compte des exigences qui ont été définies précédemment. Le Smart-hub est générique et peut être utilisé dans n'importe quel domaine. Il couvre à la fois les portées IT et OT. Il est aussi modulaire au niveau de l'interopérabilité. Ces caractéristiques existent déjà dans certains systèmes existants. Cependant, la caractéristique agnostique est l'un des principaux gains de notre système. En fait, le système est agnostique à tous les niveaux d'interopérabilité et n'impose aucune norme technique ou standard aux systèmes externes. En plus, le système est extensible à tous les niveaux d'interopérabilité. L'approche de Smart-hub peux être vue comme un jeu de lego ou les plug-ins font partie du cadre pour gérer les différents niveaux d'interopérabilité. Ces plug-ins peuvent être échangés, modifiés, ajoutés selon l'environnement. Ces plug-ins peuvent être réutilisés sans réécrire le code ("write once and reuse multiple times" [Khan et al., 2013]). En effet, la réutilisation, qui sert de base à CBSD (Component Based Software Development en anglais), joue un rôle important dans notre architecture. Il existe plusieurs définitions de la réutilisation. Selon [Krueger, 1992], la réutilisation est le processus de création de systèmes logiciels à partir de logiciels existants plutôt que de les construire à partir de zéro. [Morisio

et al., 2002] définit la réutilisation de logiciels comme la pratique systématique du développement de logiciels à partir d'un stock de blocs de construction, afin que les similarités des exigences et/ou de l'architecture entre les applications puissent être exploitées pour obtenir des bénéfices substantiels en productivité, qualité et performance. Une autre définition de la réutilisation est la capacité d'un composant logiciel précédemment développé à être réutilisé ou utilisé de manière répétée en partie ou entièrement, avec ou sans modification [Cooper, 1994]. Cette dernière définition est proche de ce que l'on entend par réutilisation de logiciels dans ce manuscrit. Dans CBSD, la réutilisation de logiciels peut être classée selon deux perspectives : développement avec réutilisation et développement pour réutilisation [Ramachandran, 2005]. Le premier traite de la façon dont les composants existants peuvent être réutilisés dans des applications existantes et nouvelles, par exemple, la réutilisation du code, des algorithmes, des bibliothèques, des composants modifiables, etc. Tandis que le dernier se réfère à la généralisation systématique des composants logiciels pour une réutilisation ultérieure. C'est-à-dire sélectionner un composant et le réutiliser tel quel.

Le Smart-hub est développé avec le point de vue du développement pour la réutilisation qui est un résultat de la modularité et de l'extensibilité. Les plug-ins adressant tous les composants d'interopérabilité peuvent être réutilisés pour tous les systèmes partageant les mêmes mécanismes de communication, syntaxe, sémantique. Des exemples de plug-ins réutilisables ont été vus dans le cas d'étude GONTRAND où quelques plug-ins de notre exemple fil rouge comme OPC UA, FileSystem, etc. ont été réutilisés pour les systèmes de GONTRAND.

Au niveau des performances du Smart-hub, la réutilisation est un facteur clé pour améliorer la productivité et la qualité des logiciels en général [Freeman, 1987] qui apportent également les mêmes avantages pour le Smart-hub. Cela se traduit par une amélioration de la mise sur le marché et le coût tel que mené par de nombreuses études [Griss, 1993, Lim, 1994, Mohagheghi et al., 2004, Poulin et al., 1993]. **Il faudrait créer un benchmark pour les différentes approches d'interopérabilité et le Smart-hub.** Cela reviendrait à réaliser toutes les approches : les normes, les middlewares, les solutions ad hoc. Nous n'avons pas réalisé un tel scénario pour valider la performance de réutilisation. C'est l'une de nos perspectives de travail futur. Un autre facteur de performance est la performance d'opération (Reponse time, availabilty, etc.) de Smart-hub. En fait, la performance d'opération a été moins étudiée dans notre recherche, car le principal problème était l'interopérabilité. La comparaison quantitative de performance entre les différentes solutions est réservée aux travaux futurs.

6.1.4 Validation théorique de la performance

Alors que la validation empirique de la performance se limite aux exemples utilisés dans les expérimentations, la validation théorique de la performance consiste à établir la confiance dans la généralité du résultat scientifique et admettre que le résultat scientifique est utile au-delà des cas d'usages étudiés. D'abord, nous nous appuyons sur l'objectif du Smart-hub : il

sert à promouvoir l'interopérabilité entre les systèmes hétérogènes en temps réel de façon générique, modulaire, agnostique et extensible.

La solution Smart-hub a été développée en tenant compte des exigences et des caractéristiques qui ont été définies précédemment dans le chapitre 1 et qui ont été revisitées dans la section de la validation théorique de la structure (cf. section 6.1.1). Ces caractéristiques sont utilisées pour évaluer la pertinence des cas d'usage dans la validation empirique de la structure. Les exemples contiennent des systèmes hétérogènes qui doivent échanger les données en temps réel chacun avec son propre mécanisme de communication, syntaxe et sémantique. Il faut aussi traiter les données et appliquer les règles métier/opérationnelle afin de mettre a disposition les données pour les DC. Ces exemples ont été soumis à plusieurs itérations pour modification à différents niveaux d'interopérabilité afin d'illustrer la réutilisation, la modularité et l'extensibilité de la solution. Ensuite, une validation empirique de la performance a été introduite pour mettre en évidence la solution en comparaison avec l'état de l'art en utilisant des exemples et des cas d'études. Même si les exemples et les cas d'études sont dans l'environnement de gaz, la solution peut être utilisée dans n'importe quel domaine où l'interopérabilité doit être abordée. Par exemple, il y a plusieurs systèmes hétérogènes dans les environnements de l'IdO, du transport, de la production, etc. Ils ont également différents mécanismes de communication, syntaxe et sémantique. De la même façon que pour le réseau de gaz, le Smart-hub peut-être utilisé avec les plug-ins pour tel environnement différent. Nous pouvons donc conclure que la performance de notre méthode est théoriquement validée.

Toutefois, nous sommes conscients que la solution du Smart-hub présente certaines limites. Le Smart-hub a été capable d'intégrer des solutions SOA qui acquièrent des données à partir d'un seul service, cependant, il n'a pas été testé pour intégrer des solutions SOA avec plusieurs services. Cela est dû au fait que le Smart-hub est orienté sur les données et non sur le service, c'est-à-dire que le point central de l'échange de données est le référentiel de données centrales et non les services. La capacité du Smart-hub pour gérer les données massives n'a pas été testée. Cependant, l'architecture Smart-hub fournit une méthodologie qui peut être mise en œuvre en utilisant des techniques plus efficaces. Le Smart-hub ne gère pas l'historique de données qui est l'un des objectifs du travail futur. Nous proposons l'approche de gestion de versions afin de gérer l'historique de données. Les aspects de sécurité, qui sont très essentiels pour tout système, ne sont pas abordés dans ce travail. Ils sont réservés aux travaux futurs.

Chapitre 7

Conclusion générale et Perspectives

Conclusion générale

La révolution numérique et industrielle a fortement impacté notre société et notre économie. Aujourd'hui, tous les systèmes, logiciels et objets, qui peuvent être connectés grâce à l'internet, peuvent fonctionner en autonomie et indépendamment. La dernière révolution industrielle souligne la nécessité de promouvoir la communication et l'échange de données entre tous les systèmes afin d'accroître la productivité de l'entreprise. Cela peut être réalisé si et seulement si les systèmes sont interopérables. L'interopérabilité est facile à réaliser dans un monde de systèmes homogènes uniques. Cependant, aujourd'hui tous les systèmes sont hétérogènes en raison de différents mécanismes de communication et interactions, différents formats et sémantiques de données y compris les règles métiers/opérationnelles.

Les travaux de cette thèse cherchent à répondre à la question de recherche suivante : comment aider les développeurs à assurer l'interopérabilité entre des systèmes hétérogènes en temps réel de manière efficace et moins cher, sans imposer une norme/technologie/domaine? Afin de répondre à cette question, nous avons effectué d'abord un état de l'art sur des approches existantes. Nous avons trouvé qu'aucune approche existante n'est parfaitement adaptée à l'interopérabilité sans imposer une norme/technologie/domaine. Pour atteindre notre objectif, nous supposons que les concepts de la séparation de préoccupations et de l'ingénierie dirigée par les modèles peuvent promouvoir l'interopérabilité de façon agnostique. Cela conduira également à favoriser la réutilisabilité et de l'extensibilité, ce qui entraînera à son tour un meilleur coût et un meilleur temps de développement pour la solution d'interopérabilité.

Nous avons alors proposé une architecture afin de promouvoir l'interopérabilité entre les systèmes hétérogènes sans imposer une norme/technologie/domaine. Cette architecture repose principalement sur la séparation des préoccupations et la décomposition fonctionnelle afin de parvenir à la modularité. Cette modularité a été réalisée via le paradigme de développent "composent based software développent/engineering (CBSD ou CBSE)". Nous

avons proposé une architecture en six composants (quatre composants horizontaux et deux composants verticaux) qui traite les différents niveaux d'interopérabilité. Elle agrège les données de plusieurs systèmes (producteurs de données - DP) et les rend disponibles pour des systèmes externes (consommateurs de données - DC).

Afin d'interagir avec chaque système sans le modifier, nous avons ajouté les concepts de plug-in aux composants. Cela permet aux développeurs d'étendre les composants principaux aux différents niveaux d'interopérabilité ce qui rend notre architecture complètement extensible.

Ensuite, nous avons introduit le cadre de l'architecture en trois briques. La brique de l'interopérabilité et la modélisation qui repose principalement sur les concepts de l'ingénierie dirigée par les modèles pour gérer les différents niveaux d'interopérabilité. La brique d'orchestration gère la réception des données provenant de DP vers notre système Smart-hub et les envoie vers les systèmes DC selon les différentes options d'interaction en temps réel en reposant sur les concepts de la programmation concurrente. La dernière brique gère la configuration de notre système pour charger les plug-ins et interagir avec les systèmes externes.

Nous avons validé notre système Smart-hub au travers de quelques cas d'étude simples et industriels. Ensuite, nous avons validé notre système en applicant le cadre du carré de validation de façon à renforcer la confiance dans l'utilité de notre système. De façon générale, nous avons trouvé que notre système est agnostique et générique car il n'impose aucune norme/technologie/domaine pour les systèmes externes. Il est modulaire et peut être étendu via des plug-ins aux différents composants grâce à la séparation de préoccupations. Cela conduit à des plug-ins réutilisables, qui à son tour conduisent à réduire le coût et le temps de développement. Notre contribution est sujette à quelques limites plus ou moins difficiles à repousser qui sont discutées à la fin de chapitre 6.

Perspectives de nos travaux

Les travaux de cette thèse forment un socle qui ouvre des perspectives intéressantes, à plus ou moins long terme.

- L'interaction avec les différents services d'un système SOA en tant qu'un seul système.

  Le Smart-hub interagit avec des solutions SOA en acquérant les données à partir d'un seul service. Autrement dit, chaque service est considéré comme un système (DP ou DC). En effet, le Smart-hub est orienté sur les données et non sur les services, c'est-à-dire que le point central de l'échange de données est le référentiel de données central et non les services. Il serait intéressant d'étudier la capacité de Smart-hub d'interagir avec plusieurs services en considérant que ces services font partie du même système. Cela imposera les besoins de réviser l'architecture afin d'exécuter le processus d'acquisition et de génération pour chaque service de système.

- La performance du Smart-hub pour gérer les données massives.

  Le Smart-hub a été testé dans certains cas d'études avec un nombre limité de systèmes et avec une fréquence de changement limitée. Cependant, dans un scénario opérationnel pour l'industrie, il faut intégrer d'une part un grand nombre de systèmes et d'autre part une fréquence de changement rapide. Il est indispensable d'étudier la performance d'opération du Smart-hub pour gérer tels scénarios. Cela impose de tester tous les aspects des performances, tels que la fiabilité, l'évolutivité, la disponibilité, le temps de réponse, etc.

  Certaines améliorations peuvent inclure l'utilisation de techniques d'architecture parallèle plus puissantes (CPU et GPU) ; tester des technologies alternatives pour gérer les ressources partagées (le référentiel central dans notre cas), en particulier pour la mise à jour du référentiel central ; intégrer des solutions Big Data en temps réel, etc.

- Gérer l'historique de données.

  Le Smart-hub se comporte comme un passage de données et ne contient que les dernières données acquises d'un système. Il sera utile de manipuler l'historique des données. En fait, certains consommateurs de données imposent la nécessité de créer de nouvelles connaissances sur la base des données collectées. Nous proposons l'approche de gestion de versions afin de gérer l'historique de données. Il pourrait aussi être réalisé via des approches classiques d'utilisation de base de données relationnelle ou graphe. Il serait intéressant d'étudier les avantages et les inconvénients de chaque approche et de sélectionner l'approche la plus appropriée qui convient à nos exigences.

- Gérer les aspects de sécurité.

  Le Smart-hub interagit avec divers systèmes qui traitent des données sensibles et importantes. Nous avons souligné le fait que notre système ne gère pas la sécurité. Nous proposons d'ajouter une composante verticale de sécurité à l'architecture. Ce composant doit garantir les exigences de sécurité telles que l'authentification, l'autorisation, le cryptage, etc. Il serait intéressant d'étudier les différentes solutions techniques pour gérer ces exigences de sécurité. Il est important de s'assurer que l'utilisation du Smart-hub n'expose pas les données et les systèmes externes aux menaces et aux vulnérabilités.

- Le chargement automatique des plug-ins (Plug and Play).

  Dans la version actuelle du système, l'utilisateur du Smart-hub le configure manuellement pour charger les plug-ins appropriés pour interagir avec chaque système externe. L'une des idées de recherche est de permettre au Smart-hub de détecter automatiquement les plug-ins appropriés avec une intervention minimale de l'utilisateur Smart-hub.

- L'application de notre architecture sur d'autres solutions techniques.

  Nous avons réalisé l'architecture générale, les briques d'interopérabilité et la modélisation spécifique en utilisant les concepts et les techniques de l'ingénierie dirigée par les modèles. Notre architecture conceptuelle fournit une méthodologie indépendante des solutions techniques spécifiques (cf. chapitre 3). Il serait intéressant d'étudier d'autres solutions techniques pour réaliser notre architecture conceptuelle et comparer leurs avantages et inconvénients.

- Créer un benchmark pour les différentes approches d'interopérabilité et le Smart-hub.

  Il faudrait créer un benchmark pour les différentes approches d'interopérabilité et le Smart-hub. Cela reviendrait à comparer toutes les approches : les normes, les middlewares, les solutions ad hoc dans les contextes et scénarios différents.

Bibliographie

IEEE standard computer dictionary : A compilation of ieee standard computer glossaries. *IEEE Std 610*, pages 1–217, Jan 1991. doi: 10.1109/IEEESTD.1991.106963. 24

IEEE standard for software verification and validation. *IEEE Std 1012-1998*, pages 1–80, July 1998. doi: 10.1109/IEEESTD.1998.87820. 125

ISO/IEC standard for systems and software engineering - recommended practice for architectural description of software-intensive systems. *ISO/IEC 42010 IEEE Std 1471-2000 First edition 2007-07-15*, pages c1–24, July 2007. doi: 10.1109/IEEESTD.2007.386501. 72

Microsoft azure iot reference architecture. Technical report, Microsoft Corporation, 2016. 46, 48

Swarup Acharya, Michael Franklin, and Stanley Zdonik. Balancing push and pull for data broadcast. *SIGMOD Rec.*, 26(2) :183–194, June 1997. ISSN 0163-5808. doi: 10.1145/253262.253293. URL http://doi.acm.org/10.1145/253262.253293. 29

D. Aksoy and M. Franklin. R times ;w : a scheduling approach for large-scale on-demand data broadcast. *IEEE/ACM Transactions on Networking*, 7(6) :846–860, Dec 1999. ISSN 1063-6692. doi: 10.1109/90.811450. 29

Mohammed M. Alani. *Guide to OSI and TCP/IP Models*. Springer Publishing Company, Incorporated, 2014. ISBN 3319051512, 9783319051512. 25

G. F. Anastasi, E. Bini, A. Romano, and G. Lipari. A service-oriented architecture for qos configuration and management of wireless sensor networks. In *2010 IEEE 15th Conference on Emerging Technologies Factory Automation (ETFA 2010)*, pages 1–8, Sept 2010. doi: 10.1109/ETFA.2010.5641336. 52

Panayiotis Andreou. Towards a network-aware middleware for wireless sensor networks. 2011. 46

Gregory R. Andrews. *Concurrent Programming : Principles and Practice*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1991. ISBN 0-8053-0086-4. 98

ECMA General Assembly. Ecmascript language specification. *Standard ECMA-262,*, 1999. 30

Edgardo Avilés-López and J. Antonio García-Macías. Tinysoa : a service-oriented architecture for wireless sensor networks. *Service Oriented Computing and Applications*, 3(2) :99–108, Jun 2009. ISSN 1863-2394. doi: 10.1007/s11761-009-0043-x. URL https://doi.org/10.1007/s11761-009-0043-x. 52

Andrew Banks and Rahul Gupta. Mqtt version 3.1. 1. *OASIS standard*, 2014. 26, 35

A Barth, Emmanuel Caillaud, and Bertrand Rose. How to validate research in engineering design? 2 :1–11, 01 2011a. 126

A Barth, Emmanuel Caillaud, and Bertrand Rose. How to validate research in engineering design? 2 :1–11, 01 2011b. 125

Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice (3rd Edition) (SEI Series in Software Engineering)*. Addison-Wesley Professional, 2012. ISBN 0321815734. 72

Alessandro Bassi, Martin Bauer, Martin Fiedler, Thorsten Kramp, Rob van Kranenburg, Sebastian Lange, and Stefan Meissner, editors. *Enabling Things to Talk*. Springer Nature, 2013. doi: 10.1007/978-3-642-40403-0. URL https://doi.org/10.1007%2F978-3-642-40403-0. 42, 53

L. Bastida, A. Berreteaga, and I. Cañadas. Adopting service oriented architectures made simple. In Kai Mertins, Rainer Ruggaber, Keith Popplewell, and Xiaofei Xu, editors, *Enterprise Interoperability III*, pages 221–232, London, 2008. Springer London. ISBN 978-1-84800-221-0. 32

Philip A. Bernstein. Middleware : A model for distributed system services. *Commun. ACM*, 39 (2) :86–98, February 1996. ISSN 0001-0782. doi: 10.1145/230798.230809. URL http://doi.acm.org/10.1145/230798.230809. 45

A. J. Berre, B. Elvesæter, N. Figay, C. Guglielmina, S. G. Johnsen, D. Karlsen, T. Knothe, and S. Lippe. The athena interoperability framework. In Ricardo J. Gonçalves, Jörg P. Müller, Kai Mertins, and Martin Zelm, editors, *Enterprise Interoperability II*, pages 569–580, London, 2007. Springer London. ISBN 978-1-84628-858-6. 63

Jean Bézivin. On the unification power of models. *Software & Systems Modeling*, 4(2) :171–188, 2005. ISSN 1619-1374. doi: 10.1007/s10270-005-0079-0. URL http://dx.doi.org/10.1007/s10270-005-0079-0. 56

Jean Bézivin. *Model Driven Engineering : An Emerging Technical Space*, pages 36–64. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-46235-4. doi: 10.1007/11877028_2. URL https://doi.org/10.1007/11877028_2. 57, 85

Thibaut Bidet-Mayer and Noé Ciet. *L'industrie du futur : une compétition mondiale*. La Fabrique de l'Industrie, 2016. 9

Thomas Bittner, Maureen Donnelly, and Stephan Winter. Ontology and semantic interoperability. *Large-scale 3D data integration : Challenges and Opportunities*, pages 139–160, 2005. 33

Michael Blackstock and Rodger Lea. IoT mashups with the WoTKit. In *2012 3rd IEEE International Conference on the Internet of Things*. Institute of Electrical and Electronics Engineers (IEEE), oct 2012. doi: 10.1109/iot.2012.6402318. URL https://doi.org/10.1109%2Fiot.2012.6402318. 52

Jaap Bloem, Menno Van Doorn, Sander Duivestein, David Excoffier, René Maas, and Erik Van Ommeren. The fourth industrial revolution. *Things Tighten*, 2014. 8

Lossan Bondé, Pierre Boulet, and Jean-Luc Dekeyser. *Traceability and Interoperability at Different Levels of Abstraction in Model-Driven Engineering*, pages 263–276. Springer Netherlands, Dordrecht, 2006. ISBN 978-1-4020-4998-9. doi: 10.1007/978-1-4020-4998-9_15. URL https://doi.org/10.1007/978-1-4020-4998-9_15. 64

Philippe Bonnet, Johannes Gehrke, and Praveen Seshadri. *Towards Sensor Database Systems*, pages 3–14. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001. ISBN 978-3-540-44498-5. doi: 10.1007/3-540-44498-X_1. URL https://doi.org/10.1007/3-540-44498-X_1. 46

Grady Booch, Alan W Brown, Sridhar Iyengar, James Rumbaugh, and Bran Selic. An mda manifesto. *Business Process Trends/MDA Journal*, 2004. 57

Pruet Boonma and Junichi Suzuki. Tinydds : An interoperable and configurable. *Principles and applications of distributed event-based systems*, page 206, 2010. 45

Carsten Bormann, Angelo P. Castellani, and Zach Shelby. CoAP : An application protocol for billions of tiny internet nodes. *IEEE Internet Computing*, 16(2) :62–67, mar 2012. doi: 10.1109/mic.2012.29. URL https://doi.org/10.1109%2Fmic.2012.29. 26, 35

Jean Pierre Bourey, Reyes Grangel, Yves Ducq, Arne Berre, M. Bertoni, Fulvio D'Antonio, Nicolas Daclin, Guy Doumeingts, Martine Grandin-Dubost, Kostas Kalampoukas, and Stilios Pantelopoulos. Report on Model Driven Interoperability, April 2007. Deliverable TG2.3. 63

J Bradley, J Barbier, and D Handler. Cisco white paper : Embracing the internet of everything to capture your share of $14.4 trillion, 2013. 9

Tim Bray. The javascript object notation (json) data interchange format. 2014. 30

Tim Bray, Jean Paoli, C Michael Sperberg-McQueen, Eve Maler, and François Yergeau. Extensible markup language (xml). *World Wide Web Journal*, 2(4) :27–66, 1997. 29, 30, 35

A. W. Brown and K. C. Wallnau. The current state of cbse. *IEEE Software*, 15(5) :37–46, Sep 1998. ISSN 0740-7459. doi: 10.1109/52.714622. 72

David Budgen. *Software Design (2nd Edition)*. Addison-Wesley, 2003. ISBN 0201722194. URL https://www.amazon.com/Software-Design-2nd-David-Budgen/dp/0201722194?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0201722194. 69

M. Caporuscio, P. G. Raverdy, and V. Issarny. ubisoap : A service-oriented middleware for ubiquitous networking. *IEEE Transactions on Services Computing*, 5(1) :86–98, Jan 2012. ISSN 1939-1374. doi: 10.1109/TSC.2010.60. 52

CENELEC CEN and ETSI. Smart grid reference architecture. Technical report, November 2012a. ix, 38, 39

CENELEC CEN and ETSI. First set of standards. Technical report, November 2012b. 38

CEN-CENELEC-ETSI and Smart Grid Coordination. Smart grid reference architecture. Technical report, European Committee for Standardization, Brussels, Belgium, nov 2012. 9, 37, 53

Werner Ceusters and Barry Smith. Semantic interoperability in healthcare. state of the art in the us. a position paper with background materials. 2010. 32

James Clark et al. Xsl transformations (xslt). *World Wide Web Consortium (W3C)*, page 103, 1999. 32, 35

Benoît Combemale. Ingénierie Dirigée par les Modèles (IDM) – État de l'art. working paper or preprint, 2008. URL https://hal.archives-ouvertes.fr/hal-00371565. x, 60, 86

Jack Cooper. Reuse-the business implications. *Marciniak Mar94*, page 1071, 1994. 132

Iván Corredor, José F. Martínez, Miguel S. Familiar, and Lourdes López. Knowledge-aware and service-oriented middleware for deploying pervasive services. *Journal of Network and Computer Applications*, 35(2) :562 – 576, 2012. ISSN 1084-8045. doi: https://doi.org/10.1016/j.jnca.2011.05.009. URL http://www.sciencedirect.com/science/article/pii/S1084804511001111. Simulation and Testbeds. 52

P. Costa, G. Coulson, R. Gold, M. Lad, C. Mascolo, L. Mottola, G. P. Picco, T. Sivaharan, N. Weerasinghe, and S. Zachariadis. The runes middleware for networked embedded systems and its application in a disaster management scenario. In *Fifth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom'07)*, pages 69–78, March 2007. doi: 10.1109/PERCOM.2007.36. 45

GridWise Architectural Council. Gridwise interoperability context-setting framework. *Smart Grids Interoperability*, pages 1–52, 2008. 37, 52, 53

Ivica Crnkovic. Component-based software engineering — new challenges in software development. *Software Focus*, 2(4) :127–133, 2001. ISSN 1529-7950. doi: 10.1002/swf.45. URL http://dx.doi.org/10.1002/swf.45. 72

Douglas Crockford. The application/json media type for javascript object notation (json). 2006. 29, 30, 35

K. Czarnecki and S. Helsen. Feature-based survey of model transformation approaches. *IBM Syst. J.*, 45(3) :621–645, July 2006. ISSN 0018-8670. doi: 10.1147/sj.453.0621. URL http://dx.doi.org/10.1147/sj.453.0621. 60, 64

Marcos Didonet Del Fabro, Jean Bézivin, and Patrick Valduriez. Model-driven tool interoperability : An application in bug tracking. In Robert Meersman and Zahir Tari, editors, *On the Move to Meaningful Internet Systems 2006 : CoopIS, DOA, GADA, and ODBASE*, pages 863–881, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-48289-5. 64

R. Drath and A. Horch. Industrie 4.0 : Hit or hype? [industry forum]. *IEEE Industrial Electronics Magazine*, 8(2) :56–58, June 2014. ISSN 1932-4529. doi: 10.1109/MIE.2014.2312079. 9

Keith Duddy and Jim R. H. Steel. Overview of the modelling of the physical world (motpw) workshop at models 2012. In *Proceedings of the Modelling of the Physical World Workshop*, MOTPW '12, pages 1 :1–1 :2, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1808-2. doi: 10.1145/2491617.2491618. URL http://doi.acm.org/10.1145/2491617.2491618. 56

Sven Efftinge and Markus Völter. oaw xtext : A framework for textual dsls. In *Workshop on Modeling Symposium at Eclipse Summit*, volume 32, page 118, 2006. 62, 64, 86

Markus Eisenhauer, Peter Rosengren, and Pablo Antolin. *HYDRA : A Development Platform for Integrating Wireless Devices and Sensors into Ambient Intelligence Systems*, pages 367–373. Springer New York, New York, NY, 2010. ISBN 978-1-4419-1674-7. doi: 10.1007/978-1-4419-1674-7_36. URL https://doi.org/10.1007/978-1-4419-1674-7_36. 49

K.A. Ellis, D. Pesch, M. Klepal, M. Look, T. Greifenberg, M. Boudon, D. Kelly, and C. Upton. Information and communication technology for EPN. In *Energy Positive Neighborhoods and Smart Energy Districts*, pages 101–157. Elsevier BV, 2017. doi: 10.1016/b978-0-12-809951-3.00005-3. URL https://doi.org/10.1016%2Fb978-0-12-809951-3.00005-3. 36, 46, 50

Brian Elvesæter, Axel Hahn, Arne-Jørgen Berre, and Tor Neple. Towards an interoperability framework for model-driven development of software systems. In Dimitri Konstantas,

Jean-Paul Bourrières, Michel Léonard, and Nacer Boudjlida, editors, *Interoperability of Enterprise Software and Applications*, pages 409–420, London, 2006. Springer London. ISBN 978-1-84628-152-5. 63

Thomas Erl. *Service-oriented architecture*, volume 8. Pearson India, 2005. 43

Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM Comput. Surv.*, 35(2) :114–131, June 2003. ISSN 0360-0300. doi: 10.1145/857076.857078. URL http://doi.acm.org/10.1145/857076.857078. 29

Dave Evans. The internet of things : How the next evolution of the internet is changing everything. *CISCO white paper*, 1(2011) :1–11, 2011. 9

P. Evensen and H. Meling. Sensewrap : A service oriented middleware with sensor virtualization and self-configuration. In *2009 International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pages 261–266, Dec 2009. doi: 10.1109/ISSNIP.2009.5416827. 52

Ludger Fiege. Data dissemination. In *Encyclopedia of Database Technologies and Applications*, pages 105–109. IGI Global, 2005. doi: 10.4018/978-1-59140-560-3.ch018. URL https://doi.org/10.4018/978-1-59140-560-3.ch018. 28, 29

Chien-Liang Fok, Gruia-Catalin Roman, and Chenyang Lu. Servilla : A flexible service provisioning middleware for heterogeneous sensor networks. *Science of Computer Programming*, 77(6) :663 – 684, 2012. ISSN 0167-6423. doi: https://doi.org/10.1016/j.scico.2010.11.006. URL http://www.sciencedirect.com/science/article/pii/S0167642310002054. (1) Coordination 2009 (2) WCRE 2009. 52

Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002. ISBN 0321127420. 72

Martin Fowler, Kent Beck, John Brant, William Opdyke, and Don Roberts. *Refactoring : improving the design of existing code*. Addison-Wesley Professional, 1999. 60

NIST Framework. Roadmap for smart grid interoperability standards. *National Institute of Standards and Technology*, 2010. 53

Michael Franklin and Stan Zdonik. “data in your face” : Push technology in perspective. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, SIGMOD '98, pages 516–519, New York, NY, USA, 1998. ACM. ISBN 0-89791-995-5. doi: 10.1145/276304.276360. URL http://doi.acm.org/10.1145/276304.276360. ix, 16, 28, 29, 35

P. Freeman, editor. *Tutorial, Software Reusability*. IEEE Computer Society Press, Los Alamitos, CA, USA, 1987. ISBN 0-818-60750-5. 132

Daniel D. Frey and Clive L. Dym. Validation of design methods : lessons from medicine. *Research in Engineering Design*, 17(1) :45–57, Jun 2006. ISSN 1435-6066. doi: 10.1007/s00163-006-0016-4. URL https://doi.org/10.1007/s00163-006-0016-4. 126

Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns : Elements of Reusable Object-oriented Software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995. ISBN 0-201-63361-2. 97

Anna Gerber, Michael Lawley, Kerry Raymond, Jim Steel, and Andrew Wood. *Transformation : The Missing Link of MDA*, pages 90–105. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002. ISBN 978-3-540-45832-6. doi: 10.1007/3-540-45832-8_9. URL https://doi.org/10.1007/3-540-45832-8_9. 60

P. B. Gibbons, B. Karp, Y. Ke, S. Nath, and Srinivasan Seshan. Irisnet : an architecture for a worldwide sensor web. *IEEE Pervasive Computing*, 2(4) :22–33, Oct 2003. ISSN 1536-1268. doi: 10.1109/MPRV.2003.1251166. 46

David Glance. Multicast support for data dissemination in orbixtalk. *Data Engineering*, 51 : 29, 1996. 29

Timo Greifenberg, Markus Look, and Bernhard Rumpe. Integrating heterogeneous building and periphery data models at the district level : The nim approach. *CoRR*, abs/1412.2961, 2014. 50

M. L. Griss. Software reuse : From library to factory. *IBM Systems Journal*, 32(4) :548–566, 1993. ISSN 0018-8670. doi: 10.1147/sj.324.0548. 132

Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2) :199–220, June 1993. ISSN 1042-8143. doi: 10.1006/knac.1993.1008. URL http://dx.doi.org/10.1006/knac.1993.1008. 33

Joaquín Guillén, Javier Miranda, Juan Manuel Murillo, and Carlos Canal. Developing migratable multicloud applications based on mde and adaptation techniques. In *Proceedings of the Second Nordic Symposium on Cloud Computing & Internet Technologies*, NordiCloud '13, pages 30–37, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2307-9. doi: 10.1145/2513534.2513541. URL http://doi.acm.org/10.1145/2513534.2513541. 64

Martin Hankel and Bosch Rexroth. The reference architectural model industrie 4.0 (rami 4.0). *ZVEI*, 2015. 38, 53

Anthony Wiles Hans van der Veer. Etsi white paper no.3 achieving technical interoperability - the etsi approach. Technical report, European Telecommunications Standards Institut, France, april 2008. 24

T. Hasiotis, G. Alyfantis, V. Tsetsos, O. Sekkas, and S. Hadjiefthymiades. Sensation : a middleware integration platform for pervasive applications in wireless sensor networks. In *Proceeedings of the Second European Workshop on Wireless Sensor Networks, 2005.*, pages 366–377, Jan 2005. doi: 10.1109/EWSN.2005.1462028. 46

Wilhelm Hasselbring. Information system integration. *Communications of the ACM*, 43(6) : 32–38, 2000. 1

W. He and L. D. Xu. Integration of distributed enterprise applications : A survey. *IEEE Transactions on Industrial Informatics*, 10(1) :35–42, Feb 2014. ISSN 1551-3203. doi: 10.1109/TII.2012.2189221. 1, 12, 13, 45

Johannes Helbig Henning Kagermann, Wolfgang Wahlster. Recommendations for implementing the strategic initiative industrie 4.0. Technical report, Germany, nov 2013. ix, 9

Jahanzaib Imtiaz and Juergen Jasperneite. Opc ua as an enabler for internet of things. Conference Paper, May 2013. 43

Systems Instrumentation and Automation Society. Ansi/isa-95.00.01-2010 (iec 62264-1 mod) enterprise-control system integration, 2010. URL http://www.isa-95.com. 37, 52

Olaronke Iroju, Abimbola Soriyan, Ishaya Gambo, and Janet Olaleke. Interoperability in healthcare : benefits, challenges and resolutions. *International Journal of Innovation and Applied Studies*, 3(1) :262–270, 2013. 32

S. Izza. Integration of industrial information systems : From syntactic to semantic integration approaches. *Enterp. Inf. Syst.*, 3(1) :1–57, February 2009. ISSN 1751-7575. doi: 10.1080/17517570802521163. URL http://dx.doi.org/10.1080/17517570802521163. ix, 12, 36, 37, 44

Frédéric Jouault, Freddy Allilaire, Jean Bézivin, and Ivan Kurtev. Atl : A model transformation tool. *Science of Computer Programming*, 72(1) :31 – 39, 2008. ISSN 0167-6423. doi: http://dx.doi.org/10.1016/j.scico.2007.08.002. URL http://www.sciencedirect.com/science/article/pii/S0167642308000439. Special Issue on Second issue of experimental software and toolkits (EST). 61, 86, 91

Frédéric Jouault and Ivan Kurtev. *Transforming Models with ATL*, pages 128–138. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-31781-4. doi: 10.1007/11663430_14. URL https://doi.org/10.1007/11663430_14. 61, 86, 91

Asif Khan, Md Mottahir Alam, Noor-ul Qayyum, and Usman Khan. Empirical study of an improved component based software development model using expert opinion technique. 5 :1–14, 07 2013. 131

Dimitrios S. Kolovos, Richard F. Paige, and Fiona A. C. Polack. *The Epsilon Transformation Language*, pages 46–60. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-69927-9. doi: 10.1007/978-3-540-69927-9_4. URL https://doi.org/10.1007/978-3-540-69927-9_4. 61

Charles W. Krueger. Software reuse. *ACM Comput. Surv.*, 24(2) :131–183, June 1992. ISSN 0360-0300. doi: 10.1145/130844.130856. URL http://doi.acm.org/10.1145/130844.130856. 131

Herbert Kubicek, Ralf Cimander, and Hans Jochen Scholl. *Layers of Interoperability*, pages 85–96. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-22502-4. doi: 10.1007/978-3-642-22502-4_7. URL http://dx.doi.org/10.1007/978-3-642-22502-4_7. 16, 24

Ivan Kurtev, Jean Bézivin, and Mehmet Aksit. Technological spaces : An initial appraisal. 01 2002. x, 61, 62

S. Lai, Jiannong Cao, and Yuan Zheng. Psware : A publish / subscribe middleware supporting composite event in wireless sensor network. In *2009 IEEE International Conference on Pervasive Computing and Communications*, pages 1–6, March 2009. doi: 10.1109/PERCOM.2009.4912862. 45

Brian Lavoie, Geneva Henry, and Lorcan Dempsey. A service framework for libraries. *D-lib magazine*, 12(7/8) :1082–9873, 2006. 46

Doug Lea. *Concurrent Programming in Java™ : Design Principles and Pattern, 2nd Edition*. Addison-Wesley Professional, 1999. ISBN 0201310090. 98

E. A. Lee. Cyber physical systems : Design challenges. In *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, pages 363–369, May 2008. doi: 10.1109/ISORC.2008.25. 10

Grace A. Lewis, Edwin Morris, Soumya Simanta, and Lutz Wrage. Why standards are not enough to guarantee end-to-end interoperability. In *Seventh International Conference on Composition-Based Software Systems (ICCBSS 2008)*. Institute of Electrical and Electronics Engineers (IEEE), feb 2008. doi: 10.1109/iccbss.2008.25. URL https://doi.org/10.1109%2Ficcbss.2008.25. 15, 53

C. Liebig, M. Cilia, M. Betz, and A. Buchmann. A publish/subscribe corba persistent state service prototype. In Joseph Sventek and Geoffrey Coulson, editors, *Middleware 2000*, pages 231–255, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. ISBN 978-3-540-45559-2. 29

W. C. Lim. Effects of reuse on quality, productivity, and economics. *IEEE Software*, 11(5) : 23–30, Sept 1994. ISSN 0740-7459. doi: 10.1109/52.311048. 132

H.K. Lin and J.A. Harding. A manufacturing system engineering ontology model on the semantic web for inter-enterprise collaboration. *Computers in Industry*, 58(5) :428 – 437, 2007. ISSN 0166-3615. doi: http://doi.org/10.1016/j.compind.2006.09.015. URL http://www.sciencedirect.com/science/article/pii/S0166361506001837. 33

Jeff Magee and Jeff Kramer. *Concurrency : State Models &Amp; Java Programs*. John Wiley & Sons, Inc., New York, NY, USA, 1999. ISBN 0-471-98710-7. 98

Wolfgang Mahnke, Stefan-Helmut Leitner, and Matthias Damm. *OPC Unified Architecture*. Springer Nature, 2009. doi: 10.1007/978-3-540-68899-0. URL https://doi.org/10.1007%2F978-3-540-68899-0. 12, 26, 33, 35, 43, 53, 90

Eric A. Marks and Michael Bell. *Service-Oriented Architecture (SOA) : A Planning and Implementation Guide for Business and Technology*. John Wiley & Sons, Inc., New York, NY, USA, 2006. ISBN 0471768944. 43

Nicola Bui Francois Carrez Christine Jardak Jourik De Loof Carsten Magerkurth Stefan Meissner Andreas Nettsträter Alexis Olivereau Matthias Thoma Joachim W. Walewski Julinda Stefa Alexander Salinas Martin Bauer, Mathieu Boussard. Internet of things - architecuture, deliverable d1.5 - final architectural reference model. Technical report, July 2013. 42

Marcello A Gómez Maureira, Daan Oldenhof, and Livia Teernstra. Thingspeak–an api and web service for the internet of things. *Retrieved7/11/15World WideWeb, http ://www. Mediate chnology. leiden. edu/images/uploads/docs/wt2014_ thing speak. pdf*, 2011. 52

Johannes Mayer, Ingo Melzer, and Franz Schweiggert. *Lightweight Plug-In-Based Application Development*, pages 87–102. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. ISBN 978-3-540-36557-0. doi: 10.1007/3-540-36557-5_9. URL https://doi.org/10.1007/3-540-36557-5_9. 72

Tim McCallum. Pipeline open data standard. *Gas Research Institute Report 00/0207*, 2000. 33

R. Meier and V. Cahill. Steam : event-based middleware for wireless ad hoc networks. In *Proceedings 22nd International Conference on Distributed Computing Systems Workshops*, pages 639–644, 2002a. doi: 10.1109/ICDCSW.2002.1030841. 45

R. Meier and V. Cahill. Steam : event-based middleware for wireless ad hoc networks. In *Proceedings 22nd International Conference on Distributed Computing Systems Workshops*, pages 639–644, 2002b. doi: 10.1109/ICDCSW.2002.1030841. 45

Peter M. Mell and Timothy Grance. Sp 800-145. the nist definition of cloud computing. Technical report, Gaithersburg, MD, United States, 2011. 64

Gero Mühl, Ludger Fiege, and Peter Pietzuch. *Distributed Event-Based Systems*. Springer, 2006. ISBN 3540326510. 26, 29

Prasiddha Misra. *Managing electronic resources*. Isha Books, New Delhi, 2011. ISBN 9788182055209. 44

Alexandre Moeuf, Robert Pellerin, Samir Lamouri, Simon Tamayo-Giraldo, and Rodolphe Barbaray. The industrial management of smes in the era of industry 4.0. *International Journal of Production Research*, 0(0) :1–19, 2017. doi: 10.1080/00207543.2017.1372647. URL http://dx.doi.org/10.1080/00207543.2017.1372647. 10

P. Mohagheghi, R. Conradi, O. M. Killi, and H. Schwarz. An empirical study of software reuse vs. defect-density and stability. In *Proceedings. 26th International Conference on Software Engineering*, pages 282–291, May 2004. doi: 10.1109/ICSE.2004.1317450. 132

M. Morisio, M. Ezran, and C. Tully. Success and failure factors in software reuse. *IEEE Trans. Softw. Eng.*, 28(4) :340–357, April 2002. ISSN 0098-5589. doi: 10.1109/TSE.2002.995420. URL http://dx.doi.org/10.1109/TSE.2002.995420. 131

Robert J Muller. *Database design for smarties : using UML for data modeling*. Morgan Kaufmann, 1999. 32

Mirco Musolesi, Cecilia Mascolo, and Stephen Hailes. Emma : Epidemic messaging middleware for ad hoc networks. *Personal and Ubiquitous Computing*, 10(1) :28–36, Feb 2006. ISSN 1617-4917. doi: 10.1007/s00779-005-0037-4. URL https://doi.org/10.1007/s00779-005-0037-4. 45

Stefano Nativi, John Caron, Ben Domenico, and Lorenzo Bigagli. Unidata's common data model mapping to the iso 19123 data model. *Earth Science Informatics*, 1(2) :59–78, Sep 2008. ISSN 1865-0481. doi: 10.1007/s12145-008-0011-6. URL http://dx.doi.org/10.1007/s12145-008-0011-6. ix, 33, 34, 36, 90

Steve Neely, Simon Dobson, and Paddy Nixon. Adaptive middleware for autonomic systems. *Annales Des Télécommunications*, 61(9) :1099–1118, Oct 2006. ISSN 1958-9395. doi: 10.1007/BF03219883. URL https://doi.org/10.1007/BF03219883. 45

David Notkin, Norman Hutchinson, Jan Sanislo, and Michael Schwartz. Heterogeneous computing environments : Report on the acm sigops workshop on accommodating heterogeneity. *Commun. ACM*, 30(2) :132–140, February 1987. ISSN 0001-0782. doi: 10.1145/12527.12529. URL http://doi.acm.org/10.1145/12527.12529. 15

Nurzhan Nurseitov, Michael Paulson, Randall Reynolds, and Clemente Izurieta. Comparison of JSON and XML data interchange formats : A case study. In *Proceedings of the ISCA 22nd International Conference on Computer Applications in Industry and Engineering, CAINE 2009,*

November 4-6, 2009, Hilton San Francisco Fisherman's Wharf, San Francisco, California, USA, pages 157–162, 2009. 30

OMG. *XML Metadata Interchange (XMI)*. OMG, 2007. 59

OMG. Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification, Version 1.1, January 2011. URL http://www.omg.org/spec/QVT/1.1/. 61

OMG. Common object request broker architecture (corba) specification, version 3.3, November 2012. URL http://www.omg.org/spec/CORBA/3.3. 45

OMG. OMG Meta Object Facility (MOF) Core Specification, Version 2.4.1, June 2013. URL http://www.omg.org/spec/MOF/2.4.1. 59

International Standard Organizatio. Industrial automation systems – concepts and rules for enterprise models, 1998. URL https://www.iso.org/standard/24020.html. 33, 36

Gerardo Pardo-Castellote, Bert Farabaugh, and Rick Warren. An introduction to dds and data-centric communications. *RTI, Aug*, 2005. 26

Thomas Paviot, Samir Lamouri, and Vincent Cheutet. A generic multicad/multipdm interoperability framework. *International Journal of Services Operations and Informatics*, 6(1-2) : 124–137, 2011. 33, 36

C. Perera, P. P. Jayaraman, A. Zaslavsky, P. Christen, and D. Georgakopoulos. Mosden : An internet of things middleware for resource constrained mobile devices. In *2014 47th Hawaii International Conference on System Sciences*, pages 1053–1062, Jan 2014. doi: 10.1109/HICSS.2014.137. 52

Heinz Bedenbender Martin Ehlich Ulrich Epple Martin Hankel Roland Heidel Michael Hoffmeister Haimo Huhle Bernd Kärcher Heiko Koziolek Reinhold Pichler Stefan Pollmeier Frank Schewe Armin Walter Bernd Waser Martin Wollschlaeger Peter Adolphs, Heinz Bedenbender. Status report : Reference architecture model industrie 4.0 (rami4.0). Technical report, July 2015. ix, 39, 40

Peter Robert Pietzuch. Hermes : A scalable event-based middleware. Technical report, 2004. 45

Jeffrey S. Poulin, Joseph M. Caruso, and Debera R. Hancock. The business case for software reuse. *IBM Syst. J.*, 32(4) :567–594, October 1993. ISSN 0018-8670. doi: 10.1147/sj.324.0567. URL http://dx.doi.org/10.1147/sj.324.0567. 132

Michael J. Pratt. Introduction to iso 10303—the step standard for product data exchange. *J. Comput. Info. Sci. Eng.*, 1(1) :102–103, March 2001a. ISSN 1530-9827. doi: 10.1115/1.1354995. URL http://dx.doi.org/10.1115/1.1354995. 31, 35

Michael J Pratt. Introduction to iso 10303—the step standard for product data exchange. *Journal of Computing and Information Science in Engineering*, 1(1) :102–103, 2001b. 37

W. Pree. Component-based software development-a new paradigm in software engineering? In *Proceedings of Joint 4th International Computer Science Conference and 4th Asia Pacific Software Engineering Conference*, pages 523–524, Dec 1997. doi: 10.1109/APSEC.1997.640216. 72

Muthu Ramachandran. Software reuse guidelines. *SIGSOFT Softw. Eng. Notes*, 30(3) :1–8, May 2005. ISSN 0163-5948. doi: 10.1145/1061874.1061889. URL http://doi.acm.org/10.1145/1061874.1061889. 132

Partha Pratim Ray. A survey of iot cloud platforms. *Future Computing and Informatics Journal*, 1(1) :35 – 46, 2016. ISSN 2314-7288. doi: https://doi.org/10.1016/j.fcij.2017.02.001. URL http://www.sciencedirect.com/science/article/pii/S2314728816300149. 47

M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke. Middleware for internet of things : A survey. *IEEE Internet of Things Journal*, 3(1) :70–95, Feb 2016a. ISSN 2327-4662. doi: 10.1109/JIOT.2015.2498900. 46

Mohammad Abdur Razzaque, Marija Milojevic-Jevric, Andrei Palade, and Siobhan Clarke. Middleware for internet of things : A survey. *IEEE Internet of Things Journal*, 3(1) :70–95, feb 2016b. doi: 10.1109/jiot.2015.2498900. URL https://doi.org/10.1109%2Fjiot.2015.2498900. 15, 44, 53

Admilson R. L. Ribeiro, Fabio C. S. Silva, Lilian C. Freitas, João Crisóstomo Costa, and Carlos R. Francês. Sensorbus : A middleware model for wireless sensor networks. In *Proceedings of the 3rd International IFIP/ACM Latin American Conference on Networking*, LANC '05, pages 1–9, New York, NY, USA, 2005. ACM. ISBN 1-59593-008-6. doi: 10.1145/1168117.1168119. URL http://doi.acm.org/10.1145/1168117.1168119. 45

J. Robert, S. Kubler, Y. L. Traon, and K. Främling. O-mi/o-df standards as interoperability enablers for industrial internet : A performance analysis. In *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, pages 4908–4915, Oct 2016. doi: 10.1109/IECON.2016.7793138. 33, 36, 90

S. Rohjans, K. Piech, and S. Lehnhoff. Uml-based modeling of opc ua address spaces for power systems. In *2013 IEEE International Workshop on Inteligent Energy Systems (IWIES)*, pages 209–214, Nov 2013. doi: 10.1109/IWIES.2013.6698587. 33

Romain Rouvoy, Paolo Barone, Yun Ding, Frank Eliassen, Svein Hallsteinsen, Jorge Lorenzo, Alessandro Mamelli, and Ulrich Scholz. *MUSIC : Middleware Support for Self-Adaptation in Ubiquitous and Service-Oriented Environments*, pages 164–182. Springer Berlin Heidelberg,

Berlin, Heidelberg, 2009. ISBN 978-3-642-02161-9. doi: 10.1007/978-3-642-02161-9_9. URL https://doi.org/10.1007/978-3-642-02161-9_9. 52

James Rumbaugh, Ivar Jacobson, and Grady Booch. *Unified modeling language reference manual, the*. Pearson Higher Education, 2004. 32

Michael Rys, Dongwon Lee, Ting Yu, Michael Rys, Denilson Barbosa, Ioana Manolescu, Jeffrey Xu Yu, Jayant R. Haritsa, Dan Suciu, Michael Rys, Michael Rys, Xin Luna Dong, Divesh Srivastava, Alon Halevy, Marcelo Arenas, Michael Weiss, Chengkai Li, Nathaniel Palmer, Peter M. Fischer, Yanlei Diao, Michael J. Franklin, Zachary Ives, Mounia Lalmas, Andrew Trotman, Michael Rys, Maya Ramanath, Juliana Freire, Neoklis Polyzotis, Denilson Barbosa, Philip Bohannon, Juliana Freire, Carl-Christian Kanne, Ioana Manolescu, Vasilis Vassalos, Masatoshi Yoshikawa, Christoph Koch, Laks V. S. Lakshmanan, Ioana Manolescu, Yannis Papakonstantinou, Vasilis Vassalos, Véronique Benzaken, Giuseppe Castagna, Haruo Hosoya, Benjamin C. Pierce, Stijn Vansummeren, Frank Neven, Giorgio Ghelli, Murali Mani, Jan Hidders, Jan Paredaens, Chavdar Botev, Jayavel Shanmugasundaram, Torsten Grust, H. V. Jagadish, Fatma Özcan, Cong Yu, and Bernd Amann. XML metadata interchange. In *Encyclopedia of Database Systems*, pages 3597–3597. Springer Nature, 2009. doi: 10.1007/978-0-387-39940-9_902. URL https://doi.org/10.1007%2F978-0-387-39940-9_902. 31, 35

Andreas Schumacher, Selim Erol, and Wilfried Sihn. A maturity model for assessing industry 4.0 readiness and maturity of manufacturing enterprises. *Procedia CIRP*, 52(Supplement C) : 161 – 166, 2016. ISSN 2212-8271. doi: https://doi.org/10.1016/j.procir.2016.07.040. URL http://www.sciencedirect.com/science/article/pii/S2212827116307909. The Sixth International Conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV2016). 10

Carolyn C Seepersad, Kjartan Pedersen, Jan Emblemsvåg, Reid Bailey, Janet K Allen, and Farrokh Mistree. The validation square : How does one verify and validate a design method? In *Decision Making in Engineering Design*. ASME Press, 2006. xi, 125, 126, 127

Daniel Serain. *Middleware and Enterprise Application Integration*. Springer, 2002. ISBN 185233570X. URL https://www.amazon.com/Middleware-Enterprise-Application-Integration-Daniel/dp/185233570X?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=185233570X. 44

Yakov Shafranovich. Common format and mime type for comma-separated values (csv) files. 2005. 29

Amit P. Sheth. *Changing Focus on Interoperability in Information Systems :From System, Syntax, Structure to Semantics*, pages 5–29. Springer US, Boston, MA, 1999. ISBN 978-

1-4615-5189-8. doi: 10.1007/978-1-4615-5189-8_2. URL http://dx.doi.org/10.1007/978-1-4615-5189-8_2. 16, 24

Jacques Durand Rajive Joshi Paul Didier Amine Chigani Reinier Torenbeek David Duggal Robert Martin (MITRE) Graham Bleakley Andrew King Jesus Molina Sven Schrecker Robert Lembree (Intel) Hamed Soroush Jason Garbis 95 Mark Crawford Eric Harper Kaveri Raman Shi-Wan Lin, Bradford Miller and Brian Witten. Industrial internet reference architecture. Technical report, 2015. 40

John Shirley. *Guide to Writing DCE Applications*. O'Reilly Media, 1992. ISBN 156592004X. URL https://www.amazon.com/Guide-Writing-Applications-John-Shirley/dp/156592004X?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=156592004X. 45

Thirunavukkarasu Sivaharan, Gordon Blair, and Geoff Coulson. *GREEN : A Configurable and Re-configurable Publish-Subscribe Middleware for Pervasive Computing*, pages 732–749. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. ISBN 978-3-540-32116-3. doi: 10.1007/11575771_46. URL https://doi.org/10.1007/11575771_46. 45

John Soldatos, Nikos Kefalakis, Manfred Hauswirth, Martin Serrano, Jean-Paul Calbimonte, Mehdi Riahi, Karl Aberer, Prem Prakash Jayaraman, Arkady Zaslavsky, Ivana Zarko, Lea Skorin-Kapov, and Reinhard Herzog. Openiot : Open source internet-of-things in the cloud. 9001 :13–25, 03 2015. 52

Eduardo Souto, Germano Guimarães, Glauco Vasconcelos, Mardoqueu Vieira, Nelson Rosa, Carlos Ferraz, and Judith Kelner. Mires : a publish/subscribe middleware for sensor networks. *Personal and Ubiquitous Computing*, 10(1) :37–44, Feb 2006. ISSN 1617-4917. doi: 10.1007/s00779-005-0038-3. URL https://doi.org/10.1007/s00779-005-0038-3. 45

Patrik Spiess, Stamatis Karnouskos, Dominique Guinard, Domnic Savio, Oliver Baecker, Luciana Moreira Sá de Souza, and Vlad Trifa. SOA-based integration of the internet of things in enterprise services. In *2009 IEEE International Conference on Web Services*. Institute of Electrical and Electronics Engineers (IEEE), jul 2009. doi: 10.1109/icws.2009.98. URL https://doi.org/10.1109%2Ficws.2009.98. 49

Dave Steinberg, Frank Budinsky, Ed Merks, and Marcelo Paternostro. *EMF : eclipse modeling framework*. Pearson Education, 2008. 85, 102

K. Su, J. Li, and H. Fu. Smart city and the applications. In *2011 International Conference on Electronics, Communications and Control (ICECC)*, pages 1028–1031, Sept 2011. doi: 10.1109/ICECC.2011.6066743. 9

Audie Sumaray and S. Kami Makki. A comparison of data serialization formats for optimal efficiency on a mobile platform. In *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication*, ICUIMC '12, pages 48 :1–48 :6, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1172-4. doi: 10.1145/2184751.2184810. URL http://doi.acm.org/10.1145/2184751.2184810. 30

Clemens Szyperski. *Component Software : Beyond Object-Oriented Programming*. 01 2002. ISBN 0-201-745572-0. 72

R. N. Taylor, N. Medvidovic, and E. M. Dashofy. *Software Architecture : Foundations, Theory, and Practice*. Wiley, 2009. ISBN 0470167742. 71

Microsoft Patterns & Practices Team. *Microsoft® Application Architecture Guide (Patterns & Practices)*. Microsoft Press, 2009. ISBN 073562710X. 72

Vagan Terziyan, Olena Kaykova, and Dmytro Zhovtobryukh. UbiRoad : Semantic middleware for context-aware smart road environments. In *2010 Fifth International Conference on Internet and Web Applications and Services*. Institute of Electrical and Electronics Engineers (IEEE), 2010. doi: 10.1109/iciw.2010.50. URL https://doi.org/10.1109%2Ficiw.2010.50. 15, 53

Amy J.C. Trappey, Charles V. Trappey, Usharani Hareesh Govindarajan, Allen C. Chuang, and John J. Sun. A review of essential standards and patent landscapes for the internet of things : A key enabler for industry 4.0. *Advanced Engineering Informatics*, 33(Supplement C) :208 – 229, 2017. ISSN 1474-0346. doi: https://doi.org/10.1016/j.aei.2016.11.007. URL http://www.sciencedirect.com/science/article/pii/S1474034616301471. 10

Mathias Uslar, Michael Specht, Sebastian Rohjans, Jörn Trefke, and José M González. *The Common Information Model CIM : IEC 61968/61970 and 62325-A practical introduction to the CIM*. Springer Science & Business Media, 2012. 32

Marcel Van Amstel, Steven Bosems, Ivan Kurtev, and Luís Ferreira Pires. Performance in model transformations : Experiments with atl and qvt. In *Proceedings of the 4th International Conference on Theory and Practice of Model Transformations*, ICMT'11, pages 198–212, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-21731-9. URL http://dl.acm.org/citation.cfm?id=2022007.2022021. 86

Pal Varga, Fredrik Blomstedt, Luis Lino Ferreira, Jens Eliasson, Mats Johansson, Jerker Delsing, and Iker Martínez de Soria. Making system of systems interoperable – the core components of the arrowhead framework. *Journal of Network and Computer Applications*, 81 : 85 – 95, 2017. ISSN 1084-8045. doi: http://doi.org/10.1016/j.jnca.2016.08.028. URL http://www.sciencedirect.com/science/article/pii/S1084804516301965. 51, 53

Alain Venot, Anita Burgun, and Catherine Quantin, editors. *Medical Informatics, e-Health*. Springer Paris, 2014. doi: 10.1007/978-2-8178-0478-1. URL https://doi.org/10.1007/978-2-8178-0478-1. 25

Ovidiu Vermesan, Peter Friess, Patrick Guillemin, Sergio Gusmeroli, Harald Sundmaeker, Alessandro Bassi, Ignacio Soler Jubert, Margaretha Mazura, Mark Harrison, Markus Eisenhauer, et al. Internet of things strategic research roadmap. *Internet of Things-Global Technological and Societal Trends*, 1 :9–52, 2011. 53

V. Veyber, A. Kudinov, and N. Markov. Model driven approach for oil amp ; gas information systems and applications integration. In *2010 6th Central and Eastern European Software Engineering Conference (CEE-SECR)*, pages 156–162, Oct 2010. doi: 10.1109/CEE-SECR.2010.5783168. 52

Vadim Veyber, Anton Kudinov, and Nikolay Markov. Model-driven platform for oil and gas enterprise data integration. *International Journal of Computer Applications*, 49(5), 2012. 52

Steve Vinoski. Advanced message queuing protocol. *IEEE Internet Computing*, 10(6), 2006. 26

Jens von Pilgrim, Bert Vanhooff, Immo Schulz-Gerlach, and Yolande Berbers. *Constructing and Visualizing Transformation Chains*, pages 17–32. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-69100-6. doi: 10.1007/978-3-540-69100-6_2. URL https://doi.org/10.1007/978-3-540-69100-6_2. 61

Peter Wegner. Interoperability. *ACM Comput. Surv.*, 28(1) :285–287, March 1996. ISSN 0360-0300. doi: 10.1145/234313.234424. URL http://doi.acm.org/10.1145/234313.234424. 24, 36, 37

Ann Wollrath, Roger Riggs, and Jim Waldo. A distributed object model for the javatm system. In *Proceedings of the 2Nd Conference on USENIX Conference on Object-Oriented Technologies (COOTS) - Volume 2*, COOTS'96, pages 17–17, Berkeley, CA, USA, 1996. USENIX Association. URL http://dl.acm.org/citation.cfm?id=1268049.1268066. 45

Michael A Zang and Jens G Pohl. Ontological approaches for semantic interoperability. *Collaborative Agent Design (CAD) Research Center*, page 78, 2003. 33

Hubert Zimmermann. Osi reference model–the iso model of architecture for open systems interconnection. *IEEE Transactions on communications*, 28(4) :425–432, 1980. 25

UTILISATION DE L'INGENIERIE DIRIGEE PAR LES MODELES POUR L'AGREGATION CONTINUE DE DONNEES HETEROGENES : APPLICATION A LA SUPERVISION DE RESEAUX DE GAZ

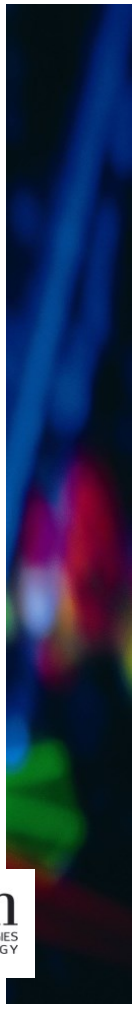
RESUME : Durant les dix dernières années, l'infrastructure informatique et l'infrastructure industrielle ont évolué de manière à passer de systèmes monolithiques à des systèmes hétérogènes, autonomes et largement distribués. Tous les systèmes ne peuvent pas coexister de manière isolée et exigent que leurs données soient partagées de manière à accroître la productivité de l'entreprise. En fait, nous progressons vers des systèmes complexes plus vastes où des millions des systèmes doivent être intégrés. Ainsi, l'exigence d'une solution d'interopérabilité peu coûteuse et rapide devient un besoin essentiel. Aujourd'hui, les solutions imposent les normes ou les middlewares pour gérer cette problématique. Cependant, ces solutions ne sont pas suffisantes et nécessitent souvent des développements ad-hoc spécifiques. Ainsi, ce travail propose l'étude et le développement d'une architecture d'interopérabilité générique, modulaire, agnostique et extensible basée sur des principes de l'architecture dirigée par les modèles et les concepts de la séparation de préoccupations. Il vise à promouvoir l'interopérabilité et l'échange de données entre les systèmes hétérogènes en temps réels sans obliger les systèmes à se conformer à des normes ou technologies spécifiques. La proposition s'applique à des cas d'usages industriels dans le contexte de réseau de distribution de gaz français. La validation théorique et empirique de notre proposition corrobore note hypothèses que l'interopérabilité entre les systèmes hétérogènes peut être atteinte en utilisant les concepts de la séparation de préoccupations et de l'ingénierie dirigée par les modèles et que le coût et le temps pour promouvoir l'interopérabilité est réduit en favorisant les caractéristiques de la réutilisabilité et de l'extensibilité.

Mots clés : interopérabilité, agrégation continue de données, middleware, ingénierie dirigée par les modèles, industrie 4.0.

MODEL-BASED INTEROPERABILITY IOT HUB FOR THE AGGREGATION OF DATA FROM HETEROGENEOUS SYSTEMS: APPLICATION TO THE SMART GAS DISTRIBUTION NETWORKS

ABSTRACT: Over the last decade, the information technology and industrial infrastructures have evolved from containing monolithic systems to heterogeneous, autonomous, and widely distributed systems. Most systems cannot coexist while completely isolated and need to share their data in order to increase business productivity. In fact, we are moving towards larger complex systems where millions of systems and applications need to be integrated. Thus, the requirement of an inexpensive and fast interoperability solution becomes an essential need. The existing solutions today impose standards or middleware to handle this issue. However, these solutions are not sufficient and often require specific ad-hoc developments. Thus, this work proposes the study and the development of a generic, modular, agnostic and extensible interoperability architecture based on modeling principles and software engineering aspects. It aims to promote interoperability and data exchange between heterogeneous systems in real time without requiring systems to comply with specific standards or technologies. The industrial use cases for this work takes place in the context of the French gas distribution network. The theoretical and empirical validation of our proposal corroborates assumptions that the interoperability between heterogeneous systems can be achieved by using the aspects of separation of concerns and model-driven engineering. The cost and time to promote the interoperability are also reduced by promoting the characteristics of re-usability and extensibility.

Keywords : interoperability, data aggregation, middleware, model-driven engineering, industry 4.0.



ARTS ET MÉTIERS ParisTech

ParisTech INSTITUT DES SCIENCES ET TECHNOLOGIES PARIS INSTITUTE OF TECHNOLOGY