Conception collaborative et aide à la décision pour les systèmes mécatroniques : Établissement d'un cadre formel.

Mouna Fradi

Thèse de doctorat

pour l'obtention du titre de

Docteur en Génie mécanique

délivré par

CY Cergy Paris Université et l'Université de Sousse

École doctorale n° 417 : Sciences et Ingénierie (SI)

Conception collaborative et aide à la décision pour les systèmes mécatroniques. Etablissement d'un cadre formel.

Thèse présentée et soutenue publiquement par

Mouna FRADI

le 14 Décembre 2021

Composition du jury :

M. Vincent Cheutet	Professeur des universités, INSA Lyon (DISP)	Président & rapporteur
M. Nizar Aifaoui	Professeur des universités, IPEIM (LGM)	Rapporteur
Mme. Claude Baron	Professeur des universités, INSA Toulouse (LAAS)	Examinatrice
M. Noureddine Ben Yahya	Professeur des universités, ENSIT (LMPE)	Examinateur
M. Jean-Yves Choley	Professeur des universités, ISAE-Supméca (Quartz)	Directeur de thèse
M. Abdelfattah Mlika	Professeur des universités, ENISo (LMS)	Co-directeur de thèse
Mme. Faïda Mhenni	Maître de conférences, ISAE-Supméca (Quartz)	Encadrante
Mme. Raoudha Gaha	Maître de conférences, UTC (Roberval)	Co-encadrante

"We keep moving forward, opening new doors, and doing new things, because we're curious and curiosity keeps leading us down new paths."

Walt Disney

Table des matières

Résumé

Les systèmes mécatroniques sont définis comme étant une intégration synergique de la mécanique, l'électronique et l'informatique assistée par des stratégies de contrôle. Cette variété de disciplines rend la tâche de développement de ces systèmes de plus en plus complexe et difficile. Pour répondre à cette complexité, les différentes disciplines d'ingénierie ont besoin de collaborer dynamiquement afin d'échanger leurs connaissances et garantir le développement d'un système cohérent et robuste.

Cependant, uniquement les connaissances qui contribuent à atteindre les objectifs du projet doivent être échangées. Ces connaissances interviennent dans le but de gérer des problèmes donnés et le risque de leur perte est considéré comme important. On appelle ces connaissances "connaissances cruciales". Durant le partage de ces connaissances, des conflits interdisciplinaires sont susceptibles de se produire et doivent être résolus pour obtenir un système cohérent. De plus, différentes alternatives ou architectures de conception pour le même système sont proposées. Ceci implique la nécessité d'une évaluation de ces alternatives selon des critères bien précis afin de converger vers l'architecture la plus satisfaisante aux regard des exigences imposées. Tenant compte de la nature pluridisciplinaire des systèmes mécatroniques et l'hétérogénéité des modèles métiers impliqués dans la collaboration, un cadre formel est nécessaire de manière à partager les connaissances cruciales. Cette formalisation permettra l'identification des connaissances cruciales et la gestion des conflits qui peuvent survenir.

Dans ce contexte, la théorie des catégories (TC) est susceptible d'être un cadre efficace de formalisation. Cette théorie a récemment fait l'objet de nombreux travaux dans le domaine de la conception collaborative et de l'ingénierie système. Motivés par le formalisme de la théorie des catégories, nous proposons une nouvelle approche pour harmoniser la collaboration durant la conception mécatronique en se basant sur les connaissances de granularité fine. Notre méthodologie, nommée *Category Theory-based Collaborative Design and Decision-making* (CaTCoDD), permet d'identifier les connaissances cruciales en s'appuyant sur la théorie des catégories. Cette théorie est utilisée dans le but de localiser ainsi que gérer les conflits. L'approche proposée permet également la comparaison entre différentes architectures du système. Finalement, un démonstrateur est mis en oeuvre sur un scénario de conception d'un actionneur électromécanique d'aileron afin de valider notre approche.

Mots clés : Conception collaborative, Systèmes mécatroniques, Gestion des connaissances, Connaissances cruciales, Gestion des conflits, Théorie des catégories, Aide à la décision.

Abstract

Mechatronic systems are defined as a synergistic integration of mechanical, electronics and computer supported by control strategies. This variety of disciplines makes the development task complex and challenging. To address this complexity, the different engineering disciplines need to collaborate dynamically to exchange knowledge and ensure the development of a coherent and robust system.

However, only knowledge that contribute to achieving the project objectives should be exchanged. Such knowledge is used to solve problems and the risk of losing it is considered important. We call the most important knowledge "crucial knowledge". During knowledge sharing, interdisciplinary conflicts are likely to occur and must be resolved to obtain a coherent system. Furthermore, different design alternatives or architectures for the same system are proposed. This implies the need to evaluate these alternatives according to specific criteria in order to find the architecture that best meets the requirements. Considereing the multidisciplinary nature of mechatronic systems and the heterogeneity of the expert models involved in the collaboration, a formal framework becomes necessary in order to unify and formalize these expert models. This formalization will allow the crucial knowledge identification and conflict management.

In this context, category theory (CT) is likely to be an effective formalization framework. This theory has recently been the subject of much work in the field of collaborative design and systems engineering. Motivated by the category theory formalism, we propose a new approach to harmonize collaboration during mechatronic design based on fine granularity knowledge. Our approach, called "Category Theory-based Collaborative Design and Decision-making" (CaTCoDD), allows the identification of crucial knowledge based on category theory. This theory is used for conflict detection and resolution. The proposed approach also deals with the evaluation of different system architectures. Finally, a demonstrator is implemented on an electromechanical actuator (EMA) of an aileron in order to validate our approach.

Keywords : Collaborative design, Mechatronic systems, Knowledge management, Crucial knowledge, Conflict management, Category theory, Decision-making.

Remerciements

Cette thèse a été réalisée dans le cadre d'une cotutelle entre le Laboratoire Quartz, Isae-Supméca Paris et le Laboratoire de Mécanique de Sousse (LMS). Elle a été effectuée sous la direction des Professeurs Jean-Yves CHOLEY et Abdelfattah MLIKA.

En tout premier lieu, je tiens à remercier mes directeurs de thèse Messieurs Jean-Yves CHOLEY et Abdelfattalh MLIKA pour m'avoir accueilli au sein de leurs équipes. La confiance que vous m'avez accordée durant ces trois années et la bienveillance avec laquelle vous m'avez laissé trouvez mon chemin, m'ont permis d'aboutir à la réussite de ce projet. Votre écoute, votre ouverture d'esprit et vos encouragements m'ont fait sentir libre et encadrée, m'offrant ainsi un environnement de travail idéal.

Mes remerciements s'adressent également à mes encadrantes mesdames Faida MHENNI et Raoudha GAHA pour avoir accepté d'encadrer ce travail. Je vous exprime ici toute ma gratitude et à ma reconnaissance pour vos conseils et vos encouragements qui m'ont permis d'avancer. Merci pour les relectures attentives de mes travaux de recherche que vous avez faites et pour la pertinence de vos remarques.

Je tiens à exprimer toute ma reconnaissance envers les Professeurs Vincent CHEUTET et Nizar AIFAOUI pour avoir accepté de participer au jury de la soutenance et d'être rapporteurs de mon mémoire de thèse.

Je tiens ensuite à témoigner toute ma gratitude envers les Professeurs Claude BARON et Noureddine BEN YAHYA pour l'honneur qu'ils m'ont fait de faire partie de mon jury de thèse.

Merci à tous les membres du laboratoire Quartz et laboratoire LMS qui ont participé de près ou de loin au bon déroulement de cette thèse. Je remercie chaleureusement mes collègues de deux équipes et mes amis pour avoir été toujours présents pour apporter du support et de l'aide.

Ces remerciements ne peuvent s'achever, sans une pensée pour mes chers parents. Je remercie sincèrement mon père Salah et ma mère Samia qui m'ont inspiré la persévérance et la détermination. Vous m'avez supporté tout au long de mes longues années d'études et vos conseils attentionnés m'ont permis de tracer la bonne route. Merci du fond du coeur, sans vous je ne serais jamais arrivée là. Merci d'avoir cru en moi !

Enfin, je réserve une place toute particulière à mes chères soeurs Samessem, Sana, Wiem, Nada et Wafa, qui ont su m'apporter amour et aide pendant toutes mes années d'études. Je vous adresse toute ma profonde reconnaissance pour m'avoir écoutée et soutenue. Merci d'être toujours là pour moi aussi bien dans les moments durs et joyeux. Rien n'aurait été possible sans vous. Merci de m'avoir offert des neveux adorables qui m'ont apporté la joie avec leurs sourires angéliques.

Mouna.

Liste des tableaux

Table des figures

Abréviations

AC	**A**rchitecture **C**onfiguration
ADMC	**A**ide à la **D**écision **M**ulti-**C**ritère
AHP	**A**nalytical **H**ierarchical **P**rocess
ANSI	**A**merican **N**ational **S**tandards **I**nstiute
AOE	**A**rchitecture **O**utput **E**ntity
ASP	**A**rchitecture **S**election **P**rocess
BDD	**B**loc **D**efinition **D**iagram
CAO	**C**onception **A**ssistée par **O**rdinateur
CaTCoDD	**Ca**tegory **T**heory-based **Co**llaborative **D**esign and **D**ecision-making
CDP	**C**ollaborative **D**esign **P**rocess
CDPPK	**C**ollaborative **D**esign **P**rocess and **P**roduct **K**nowledge
CI	**C**onsistency **I**ndex
CKIP	**C**rucial **K**nowledge **I**dentification **P**rocess
CMGVC	**C**oncept- **M**odel- **G**raph- **V**iew **C**ycle
COLIBRI	**CO**nstraint **LI**nking **BRI**dge

COTS **C**ommercial **O**ff-**T**he-**S**helf

CPS **C**yber-**P**yhsical **S**ystem

CR **C**onsistency **R**ule

Cr **C**onsistency **R**atio

CRP **C**onflict **R**esolution **P**rocess

CSP **C**onstraint **S**atisfaction **P**roblem

DC **D**ependency **C**ategory

DPK **D**esign **P**roduct **K**nowledge

DPS **D**igital **P**roduct **S**imulation

E2KS **E**ngineering **E**ntreprise **K**nowledge **S**ystem

ECIA **E**lectronic **C**omponents **I**ndustry **A**ssociation

EFFBD **E**nhanced **F**unctional **F**low **B**lock **D**iagram

EIA **E**lectronic **I**ndustries **A**lliance

ELECTRE **EL**imination **E**t **C**hoix **T**raduisant la **RE**alité

EM **E**xpert **M**odel

EMA **E**lectro-**M**echanical **A**ctuator

ETB **E**lectronic **T**hrottle **B**ody

EVAMIX **EVA**luation **MIX**ed criteria

FPCI **F**inal **P**arameter **C**ategory **I**nstance

GAMETH **G**lobal **A**nalysis **METH**odology

GUI **G**raphical **U**ser **I**nterface

ICE **I**nformation **C**ore **E**ntity

IEC **I**nternational **E**lectrotechnical **C**ommission

IS **I**ngénierie **S**ystème

ISO **I**nternational **O**rganization for **S**tandardization

KARREN **K**nowledge **A**cquisition and **R**euse for **R**obust **EN**gineering

KBE **K**nowledge-**B**ased **E**ngineering

KC **K**nowledge **C**onfiguration

KClass **K**nowledge **Class**ification

KCMethod **K**nowledge **C**onfiguration **Method**

KE **K**nowledge **E**ngineering

KM **K**nowledge **M**anagement

KPac **K**nowledge **Pac**ket

MACBETH **M**easuring **A**ttractiveness by a **C**ategorial **B**ased **E**valuation **TecH**nique

MAUT **M**ultiple **A**ttribute **U**tility **T**heory

MAVT **M**ultiple **A**ttribute **V**alue **T**heory

MBSE **M**odel-**B**ased **S**ystems **E**ngineering

MDA **M**odel-**D**riven **A**rchitecture

MDE **M**odel-**D**riven **E**ngineering

MDI **M**echatronic **D**esign **I**ndicator

MDQ **M**echatronic **D**esign **Q**uotient

MIV **M**echatronic **I**ndex **V**ector

MMP **M**echatronic **M**ulticriteria **P**rofile

MOKA **M**ethodology and tools **O**riented to **K**BE **A**pplications

MOS **M**echatronic **O**ntology **S**ystem

OMG **O**bject **M**anagement **G**roup

PC **P**arameter **C**ategory

PCI **P**arameter **C**ategory **I**nstance

PD **P**roject **D**omain

PID **P**roportionnelle **I**ntégrale **D**érivée

PK **P**roduct **K**nowledge

PLM **P**roduct **L**ifecycle **M**anagement

PROMETHEE **P**reference **R**anking **O**rganization **METH**od for **E**nrichment

PtC **P**a**t**tern **C**ategory

QVT **Q**uery **V**iew **T**ransformation

RA **R**esolution **A**ction

RI **R**andom **I**ndex

SLIM **S**ystems **LI**fecycle **M**anagement

SMART **S**imple **M**ulti-**A**ttribute and **R**ating **T**echnique

SysML **Sys**tems **M**odeling **L**anguage

TC	**T**héorie des **C**atégories
TOPSIS	**T**echnique for **O**rder **P**reference by **S**imilarity to **I**deal **S**olution
UC	**U**ser **C**onfiguration
UML	**U**nified **M**odeling **L**anguage
UPC	**U**ser **P**rocess **C**onfiguration
UPCI	**U**pdated **P**arameter **C**ategory **I**nstance
VDI	**V**erein **D**eutscher **I**ngenieure (Association des ingénieurs allemands)

Introduction générale

Contexte général

Dans le contexte industriel et économique actuel, les entreprises se trouvent devant un défi qui est l'amélioration de la qualité des produits tout en minimisant les délais et les coûts de développement [1]. Pour atteindre ces objectifs, une attention particulière est accordée à la conception collaborative, qui a remplacé le processus de conception dit séquentiel [2]. Cette méthode séquentielle n'est plus utile pour supporter la conception des systèmes complexes imposant une synergie entre différentes disciplines [3]. Étant donné que la conception des systèmes mécatroniques a eu recours à diverses disciplines, notamment la mécanique, l'électronique, l'informatique et l'automatique, une collaboration efficace est ainsi nécessaire pour converger vers un système robuste et cohérent [4]. Chaque discipline s'intéresse à un aspect particulier du système et exploite différents outils d'ingénierie. Cette diversité rend la conception mécatronique assez complexe. Pour faire face à cette complexité, l'intégration synergique de différentes disciplines a fait l'objet de nombreux travaux antérieurs. La gestion du cycle de vie des produits appelé *Product Lifecycle Management* (PLM) a été proposée afin de fournir un environnement de collaboration entre les parties prenantes d'un produit [5]. Cependant, le PLM a démontré son incapacité à supporter la complexité de la conception mécatronique [6]. Par conséquent, il y a eu recours aux modèles de connaissances pour supporter la collaboration durant la conception mécatronique [7]. Ces modèles permettent l'échange de connaissances potentiellement utiles à la collaboration [8]. Ces connaissances sont considérées comme étant cruciales pour atteindre les objectifs de conception. Cependant, l'identification de telles connaissances devient un challenge pour les concepteurs [9]. En effet, des efforts particuliers ont été consacrés au développement des méthodes d'extraction des connaissances dites cruciales. Des méthodes informelles basées sur un échange étroit entre les différentes parties prenantes d'un projet ont été proposées afin d'identifier les connaissances utiles à la collaboration [7, 10]. Néanmoins, l'échange entre diverses équipes semble difficile à organiser, particulièrement dans un environnement distribué et complexe comme celui de la conception mécatronique. Cette méthode informelle nécessite un temps considérable et entraîne souvent des risques de perte de connaissances importantes et utiles. C'est à ce niveau qu'apparaît la nécessité d'intégrer une méthode formelle dans les modèles de connaissances afin de faciliter

la tâche d'extraction des connaissances cruciales. Ces dernières feront ensuite l'objet de capitalisation qui consiste à considérer les connaissances cruciales comme un capital pour l'entreprise [11] et qui fournit ainsi à l'entreprise une mémoire collective de connaissance qui peut être employée à des fins de ré-utilisation.

Par ailleurs, durant la conception mécatronique, différents acteurs de diverses disciplines sont impliqués dans le processus de conception. En ce sens, des modèles métiers hétérogènes sont établis. Bien que ces modèles soient distincts et traitent différents points de vue du système à concevoir, l'échange et la communication sont inévitables [12]. Par conséquent, des conflits entre les modèles en question peuvent se présenter et doivent être ainsi gérés. Divers méthodes et outils ont été mis en place dans le but de localiser et résoudre les conflits [13]. Ce concept a été également abordé dans les modèles de connaissances qui supportent la collaboration et le partage des connaissances. Cependant, des solutions explicites pour détecter ainsi que gérer les conflits entre les modèles métiers ne sont pas fournies dans les supports de collaboration. Un effort particulier est ainsi nécessaire afin de résoudre les conflits au sein des modèles de connaissance.

Les défis de la conception collaborative des systèmes mécatroniques ne résident pas seulement dans l'identification des connaissances cruciales et la gestion des conflits, mais aussi dans l'évaluation et le choix d'architecture du système à concevoir. Considérant la diversité des domaines impliqués dans la conception mécatronique, différentes alternatives ou architectures peuvent être générées [14]. Une décision inappropriée dans le choix d'une architecture peut entraîner des surcoûts au niveau du temps de développement. D'où l'importance d'évaluer, en phase préliminaire de conception, les différentes alternatives afin d'éviter les boucles de corrections et réduire le temps de développement. Le choix d'architecture des systèmes mécatroniques n'a pas été abordé dans les modèles de connaissances [7]. Dans ces derniers, une seule architecture du système mécatronique a été étudiée. Ainsi, l'intégration de l'aide à la décision pour choisir l'architecture la plus appropriée est devenue un enjeu clé pour les supports de collaboration.

Ainsi, le développement d'une nouvelle méthodologie de collaboration et d'aide à la décision permettant, à la fois, l'identification formelle des connaissances cruciales, la gestion des conflits et également l'évaluation de différentes architectures, est devenu un défi important pour les industriels adoptant une démarche collaborative pour la conception des systèmes.

Objectifs scientifiques

L'objectif de recherche, au travers de cette thèse, est de développer une méthodologie générale et un cadre formel associé destinés à la collaboration et l'aide à la décision pour les systèmes mécatroniques. D'une part, cette méthodologie doit soutenir l'identification des connaissances cruciales en se référant à un cadre formel. D'autre part, notre approche doit fournir une solution pour localiser et également résoudre les conflits qui peuvent survenir durant les échanges et la collaboration.

Enfin, notre proposition doit répondre au besoin d'évaluation de différentes architectures d'un système mécatronique.

Dans le but d'atteindre ces objectifs, nous introduisons, dans un premier temps, une méthode pour extraire formellement les connaissances cruciales. L'idée réside dans la mise en place d'un cadre mathématique formel facilitant la classification des connaissances. Motivés par la base formelle puissante de la théorie des catégories, le cadre formel sera basé essentiellement sur cette théorie mathématique. Ce cadre sera étendu, dans un second temps, afin de faciliter la gestion des conflits. Seuls les conflits qui peuvent survenir entre les connaissances d'un niveau de granularité fine (paramètres et contraintes) sont traités dans cette démarche. La détection ainsi que la résolution des conflits sont faites en se basant sur des règles de cohérences et de *patterns* mettant en évidence les conflits et les incohérences qui peuvent apparaître entre les modèles métiers. Enfin, nous présentons des travaux d'intégration d'une méthode d'aide à la décision afin d'orienter le choix d'architectures du chef de projet. Cette méthode vise à classifier les différentes alternatives générées durant le processus de conception. Cette classification repose sur un ensemble de critères identifiés en fonction des objectifs du chef de projet.

En résumé, les résultats attendus de ce travail de recherche sont les suivants :

— Une méthodologie générale assurant la conception collaborative des systèmes mécatroniques en se basant sur la théorie des catégories, compatible à la fois, avec une démarche d'identification formelle des connaissances cruciales, une méthode de gestion des conflits et une approche de choix d'architectures.

— La formalisation de la méthodologie proposée avec un méta-modèle appelé *"Category Theory-based Collaborative Design and Decision-making"* (CaT-CoDD). Ainsi, ce méta-modèle mettra en évidence tous les concepts de base de la méthodologie développée.

— L'outillage de l'approche proposée sous forme d'un démonstrateur. Ce dernier doit assurer une collaboration efficace entre les différents acteurs impliqués dans le même projet de conception tout en soutenant tous les aspects de ce partage des connaissances.

Organisation du mémoire

Ce manuscrit est organisé en quatre chapitres.

Comme le tout premier stade qui précède le développement d'une méthodologie générale dédiée à la conception collaborative mécatronique consiste à définir et comprendre les concepts principaux de ce domaine, un premier chapitre est dédié à la présentation de la mécatronique ainsi que les aspects fondamentaux de la conception collaborative. Les travaux antérieurs menés dans le contexte de la conception collaborative sont dressés et analysés. A l'issue de cette analyse, nous avons identifié la problématique à laquelle nos efforts de recherche tentent de répondre.

Le deuxième chapitre, expose la méthodologie générale proposée afin d'adresser les limites des travaux antérieurs. Nous commençons ce chapitre par la présentation

des propositions établies pour aborder les axes de recherche. Les différentes étapes ainsi que les principes fondamentaux de notre méthodologie sont détaillés. Le second chapitre s'achève avec la formalisation de la méthodologie à l'aide d'un méta-modèle que nous appelons "*CaTCoDD model*" ainsi que la présentation du démonstrateur CaTCoDD.

Le troisième chapitre se focalise d'une part sur l'identification des connaissances dites cruciales et qui doivent faire l'objet de capitalisation durant la collaboration. L'approche, que nous désignons par "*Crucial Knowledge Identification Process (CKIP)*", est présentée. Nous détaillons également les différentes étapes qui nous amène à la classification des connaissances. Les différents concepts constituant l'approche CKIP sont mis en évidence. D'autre part, dans ce chapitre, une approche surnommée "*Conflict Resolution Process (CRP)*" pour la résolution des conflits est exposée. Cette proposition vise à détecter et gérer les conflits survenus entre les différents modèles métiers participant à la conception collaborative d'un système mécatronique. L'ensemble des concepts utilisés dans l'approche CRP est explicité. L'architecture globale de CRP est aussi soulignée dans cette partie.

Ayant identifié les limites des supports de collaboration en terme d'aide à la décision, démontrées dans le premier chapitre, une intégration d'une méthode d'aide à la décision dans notre méthodologie CaTCoDD est abordée dans la dernière partie du troisième chapitre. Nous désignons cette approche par "*Architecture Selection Process (ASP)*". Une description détaillée de la méthode mise en place afin d'aider le concepteur à choisir l'architecture la plus appropriée de son système est menée. Les différentes étapes de cette intégration sont aussi développées.

Dans le dernier chapitre, la méthodologie adoptée est implémentée à travers l'application à un actionneur électromécanique d'aileron. Les différents niveaux du démonstrateur réalisé à partir du modèle CaTCoDD sont présentés. Ce démonstrateur est mis en place afin de donner une vision globale et réelle de l'implémentation de la méthodologie proposée.

Finalement, une conclusion vient clore ce mémoire. Un bilan de nos objectifs ainsi que nos contributions est présenté. Les perspectives qui peuvent être envisagées afin d'améliorer ce travail de recherche sont également exposées.

Chapitre 1

Enjeux de la conception collaborative et état de l'art

Une attention particulière est apportée par les industriels aux systèmes mécatroniques où le compromis délai-qualité-coût est toujours en question. Comme notre objectif consiste à développer une méthodologie supportant la conception collaborative et le partage des connaissances, il convient de commencer par introduire les concepts en relation avec la mécatronique et la collaboration. Une étude bibliographique couvrant les supports de collaboration, l'identification des connaissances cruciales, la gestion des conflits et le choix d'architectures est détaillée. A l'issue de cette étude, le positionnement de nos contributions par rapport aux travaux antérieurs est aussi souligné.

1.1 Généralités sur les systèmes mécatroniques

Le terme "mécatronique" ou "*Mechatronics*" a été présenté pour la première fois par un ingénieur de la compagnie Yaskawa Electric Corporation [15]. Ce terme provient à la base de la concaténation de 'Méca' de Mécanique et 'tronique' de l'électronique [16]. Plusieurs définitions ont été proposées afin de définir la mécatronique. Harashima et al. ont introduit ce concept comme « L'intégration synergique de l'ingénierie mécanique avec le contrôle intelligent et l'électronique pour la conception et la fabrication des produits » [17]. Par ailleurs, une autre proposition a été présentée où la mécatronique est définie comme « Un domaine interdisciplinaire où la mécanique, les systèmes de contrôle, les systèmes électroniques ainsi que les technologies de l'informatique coopèrent » [18]. Selon la norme française NF E 01-010 (2008), la mécatronique est présentée comme une « démarche visant l'intégration en synergie de la mécanique, l'électronique, l'automatique et l'informatique dans la conception et la fabrication d'un produit en vue d'augmenter et/ou optimiser la fonctionnalité». Même si ces propositions diffèrent en quelques points, elles conduisent à une définition commune où la mécatronique est une collaboration entre la mécanique, l'électronique, le contrôle et l'informatique afin de concevoir un système synergique comme nous pouvons le voir sur la figure (1.1).

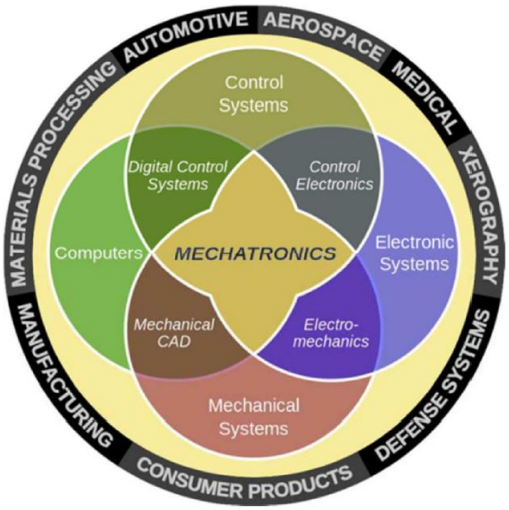


FIGURE 1.1 – Interdisciplinarité de la mécatronique et ses applications [19]

L'expansion de la mécatronique n'a cessé d'augmenter depuis l'apparition des systèmes mécatroniques aussi bien dans les industries que dans notre quotidien. Des applications variées de ces systèmes complexes sont mises en place dans divers domaines comme illustré par la figure (1.2).

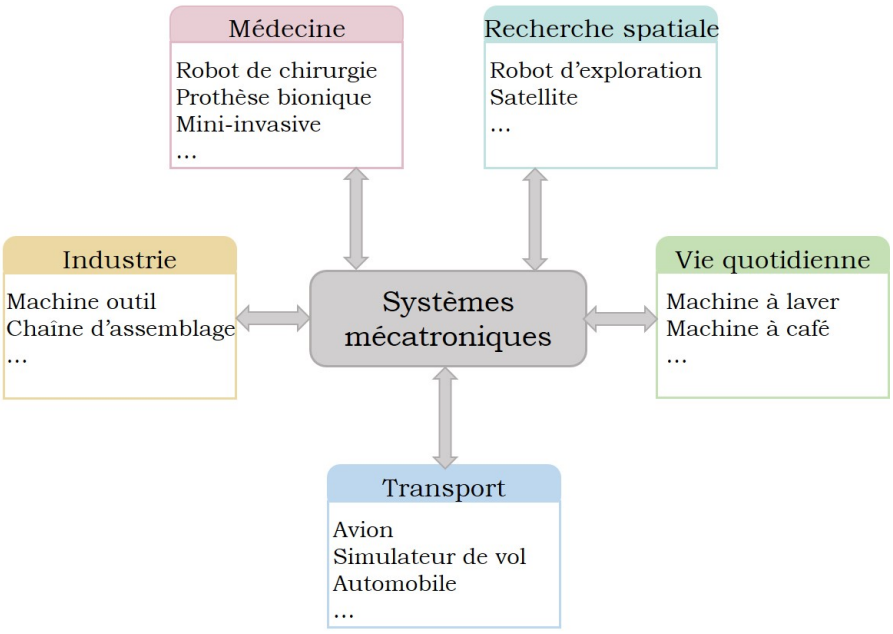


FIGURE 1.2 – Domaines d'applications des systèmes mécatroniques [20]

1.2 Processus de conception pour les systèmes mécatroniques

La conception représente une activité de création permettant à aboutir à la définition d'un produit satisfaisant aux besoins. La conception représente ainsi l'ensemble des processus et des activités soutenant le passage de l'idée de l'amélioration d'un produit existant ou d'un nouveau produit vers sa production tout en assurant sa maintenabilité et son usage [21]. En ce sens, de nombreux modèles de conception ont été proposés afin de supporter la conception mécatronique. Nous présentons dans la suite les principaux processus de conception.

1.2.1 Processus séquentiel

Différentes approches considèrent la conception des systèmes mécatroniques comme un processus séquentiel. Le modèle de Pahl et Bietz est l'un des modèles largement reconnus. Les étapes de ce modèle sont illustrées par la figure (1.3). Quatre étapes successives sont soulignées dans le modèle de Pahl et Bietz. La première étape revient à classifier la tâche de conception. Le cahier des charges fonctionnel et les spécifications techniques sont établis dans cette étape. Ceci conduit à la spécification des fonctions, contraintes et caractéristiques du produit à concevoir. Durant la deuxième étape, les besoins fonctionnels sont identifiés. Une description technique

est ensuite dressée lors de la troisième phase. A ce niveau, différentes alternatives du produit sont comparées afin de converger vers l'alternative la plus appropriée. La dernière étape de ce processus consiste à établir un dossier de définition du produit mettant en évidence des plans détails ainsi que des coûts prévisionnels.

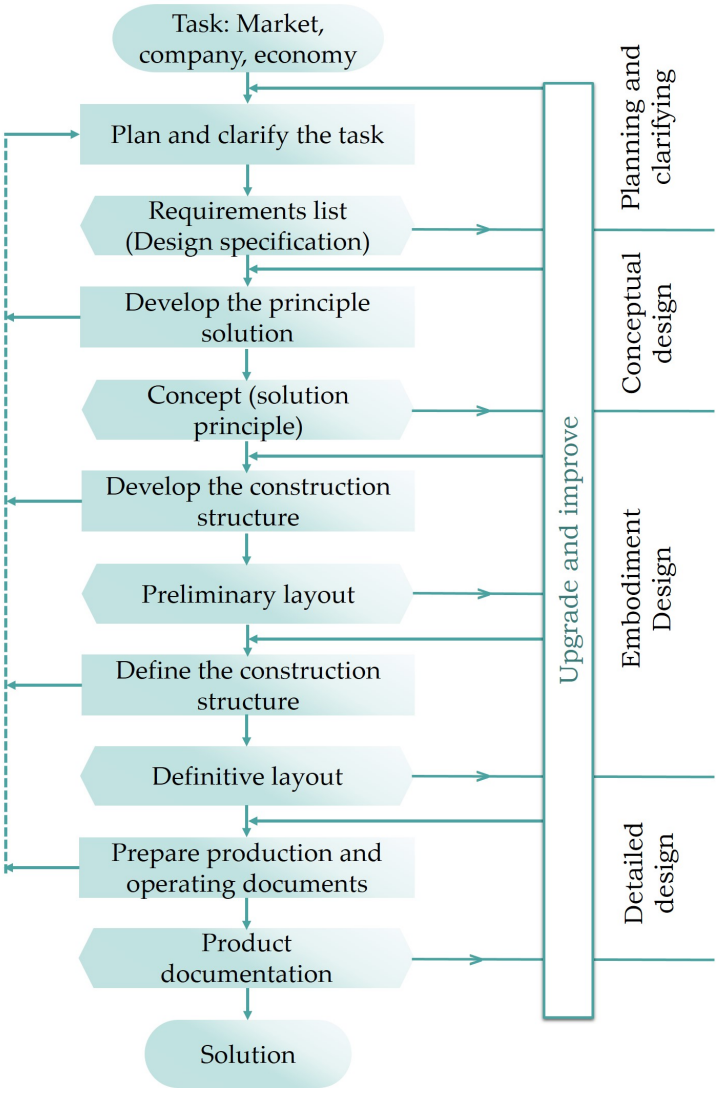


FIGURE 1.3 – Processus séquentiel de conception [22]

1.2.2 Modèle par division disciplinaire

Un autre modèle de conception fondé sur une répartition des exigences entre les parties prenantes, notamment les équipes mécaniques, logicielles et électroniques, est proposé par Aca et al. [23]. En ce sens, le travail des équipes impliquées dans la

conception est axé sur une division disciplinaire où les concepteurs travaillent sur des sous-projets. L'intégration multidisciplinaire fait l'objet d'une étape indépendante dans le processus de conception. Cependant, une vue globale sur la conception et les problèmes rencontrés n'est possible que pour un nombre limité de concepteurs [24]. La structure du modèle de conception par division disciplinaire est illustrée par la figure (1.4).

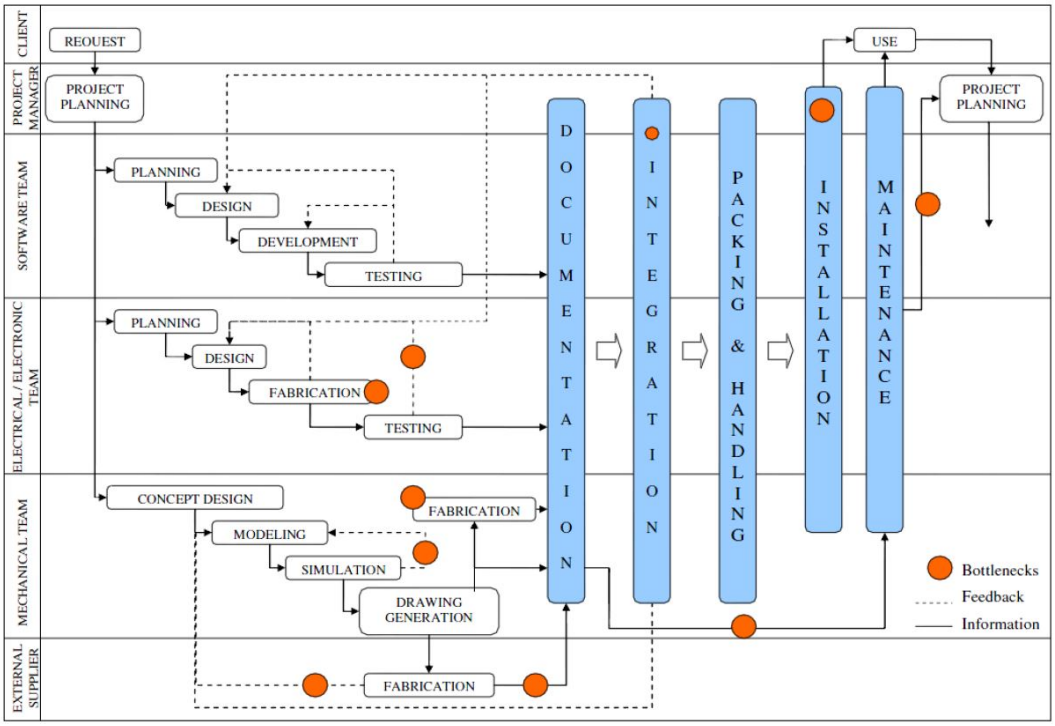


FIGURE 1.4 – Modèle de conception basé sur la division par discipline [23]

1.2.3 Cycle en V

Comme les processus basés sur une démarche séquentielle ont démontré leurs limites pour supporter la complexité de la conception mécatronique, des efforts ont été menés afin d'intégrer les activités de conception. Le cycle en V est un processus largement répandu dans la conception des produits. Ce cycle a été normalisé pour les systèmes mécatroniques par une association allemande sous forme de la directive VDI 2206 [25]. Même si la forme de ce processus est séquentielle, sa forme en « V » met en évidence les liens entre les deux phases. Les disciplines sont traitées en parallèle pour mettre l'accent sur les itérations qui se produisent au cours du cycle de conception [7]. La première phase est descendante et illustrant la décomposition du système alors que la deuxième est ascendante mettant en valeur l'intégration du

système (voir figure (1.5)). Ce cycle s'articule autour de trois étapes :

— Étape 1 : *System design* : cette étape est décomposée en deux sous-étapes. La première consiste à identifier les exigences et estimer le coût et le temps de développement [26], alors que durant la deuxième sous-étape les différentes fonctions du système sont identifiées. Plusieurs architectures peuvent être ainsi analysées et comparées afin de choisir l'architecture la plus appropriée.
— Étape 2 : *Domain-specific design* : c'est l'étape intermédiaire du cycle en « V ». Au cours de cette étape, l'architecture choisie est transformée en solutions techniques et modélisée ensuite en utilisant différents outils de modélisation [27].
— Étape 3 : *System integration* : La dernière étape consiste à effectuer des simulations dans le but de tester le système. L'intégration des différents composants a lieu dans cette étape. Finalement, les exigences définies au début du processus de conception sont validées en ayant recours à des outils de validation multidisciplinaire [28].

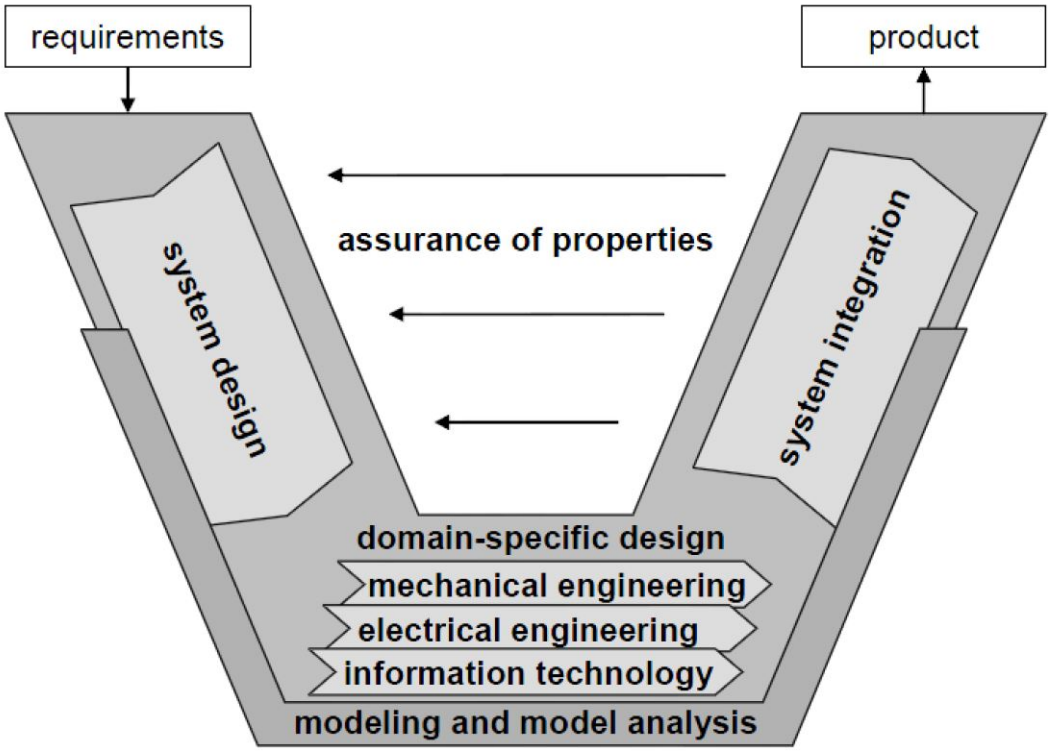


FIGURE 1.5 – Le cycle de conception en « V » selon la directive VDI 2206 [29]

Une nouvelle révision de ce modèle a été élaborée par Graessler Hentze [30] dans le but de supporter la conception mécatronique et la conception des systèmes cyber-physiques (*Cyber-Physical Systems*) (CPS) comme détaillé dans la figure (1.6).

La différence majeure entre le cycle en V et sa révision réside dans le fait que la collecte, la modification ainsi que la gestion des exigences s'effectuent tout au long de la nouvelle version du cycle en V. Par ailleurs, cette version révisée propose de fusionner le choix d'architecture avec la conception préliminaire. En ce sens, notre travail de recherche se positionne dans cette phase où le choix d'architecture se fait dès les premières phases de conception.

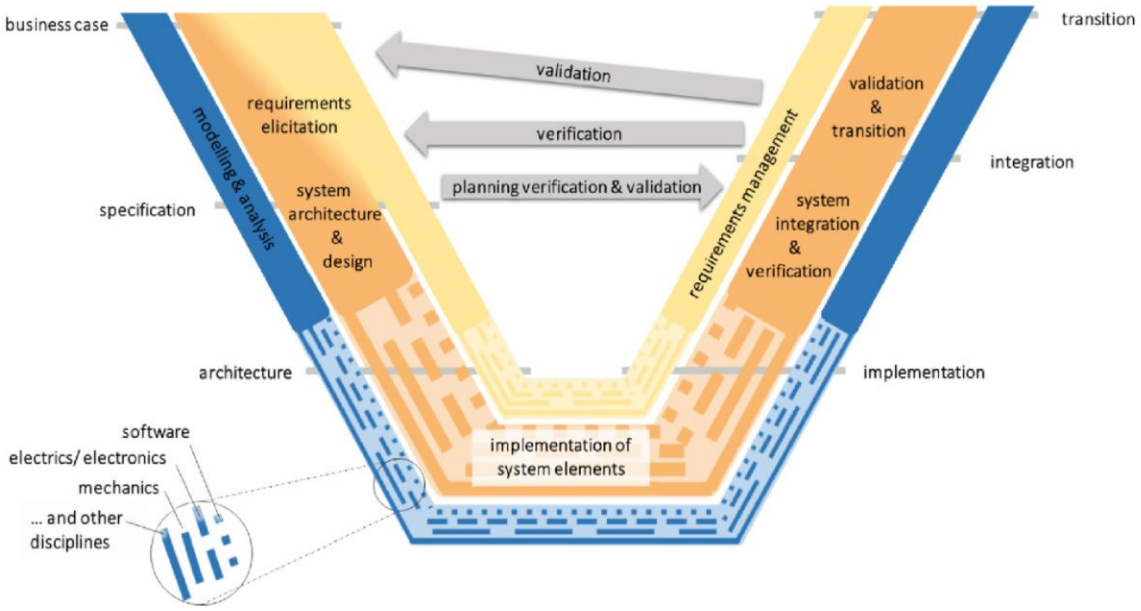


FIGURE 1.6 – La nouvelle révision du cycle de conception en « V » [30]

1.3 Enjeux de la conception collaborative et d'ingénierie Système (IS)

1.3.1 De la conception séquentielle vers la conception collaborative

Les démarches séquentielles de conception ont démontrées leur incapacité à supporter la complexité des systèmes mécatroniques vu qu'elles sont basées sur une succession hiérarchique des phases. Cet enchaînement successif ne semble pas correspondant à la réalité des pratiques industrielles où l'interaction et la collaboration entre les différents acteurs s'imposent durant certaines phases du processus de conception. Pour cette raison, il s'avère utile de rompre avec le processus séquentiel et mettre en étroite collaboration les différentes phases de conception [31]. Cette ingénierie collaborative consiste à établir les différentes activités de conception d'une

manière parallèle afin de réduire le temps de développement et améliorer la qualité des produits comme nous pouvons le constater sur la figure (1.7). Ce concept est apparu dans les années quatre-vingt sous différentes appellations telles que la conception intégrée [32], concourante [33], voire collaborative [34]. Cette conception collaborative a contribué à améliorer la flexibilité et augmenter la productivité [35].

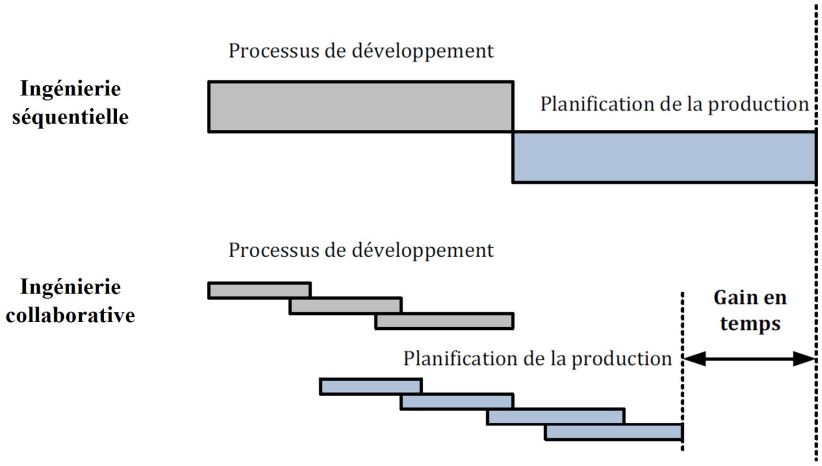


FIGURE 1.7 – De la conception séquentielle vers la conception collaborative [36]

1.3.2 L'ingénierie Système (IS)

L'ingénierie Système (IS) est une approche interdisciplinaire permettant la réalisation des systèmes tout en se concentrant sur le système global plutôt que sur ses différents composants [37]. Ce concept décrit une vue d'ensemble des processus utilisés pour concevoir les systèmes complexes tels que les systèmes mécatroniques [38]. Le concept de l'IS a fait l'objet de nombreuses normes tels que la norme ANSI/EIA 632 développée par l'association de l'industrie des composants électroniques ou *Electronic Components Industry Association* (ECIA) et adoptée ensuite par l'institut national américain de normalisation ou (*American National Standards Instiute* (ANSI) [39] et le standard ISO 15288 standardisé par la commission électrotechnique internationale ou *International Electrotechnical Commission* (IEC) [40]. Le concept de l'IS est étendu afin de capturer les informations des systèmes en utilisant des modèles. Cette approche est appelée l'ingénierie système basée sur les modèles, connue par *Model-Based Systems Engineering* (MBSE). L'approche MBSE permet d'assurer la cohérence entre les activités d'ingénierie tout en améliorant la qualité du produit final. Les modèles au niveau système sont générés avec cette approche ce qui permet de créer une base de connaissances et de favoriser l'échange entre les parties prenantes impliquées dans la conception [41].

Dans le but de supporter l'IS et l'approche MBSE, divers langages de modélisation ont été proposés tels que le langage *Enhanced Functional Flow Block Diagram* (EFFBD) et le langage SysML *(Systems Modeling Language)*. Ce denier est considéré comme le langage le plus adapté pour l'IS [42]. Basé sur le langage *Unified Modeling Language* (UML), SysML est conçu pour rassembler les méthodes et les outils utilisés en ingénierie dans un langage unique et standardisé. SysML est un langage semi-formel basé sur la modélisation graphique. Il fournit des constructions simples mais aussi puissantes pour faire face aux problèmes de l'IS [43]. Comme il est essentiellement basé sur le langage UML, SysML a hérité certains diagrammes d'UML tout en apportant ses propres diagrammes. Neuf diagrammes, résumés dans la figure (1.8), ont été définis dans ce langage. Ils sont classifiés selon le type comme illustré par la figure (1.8) : diagrammes pour représenter le comportement, les exigences et la structure du système.

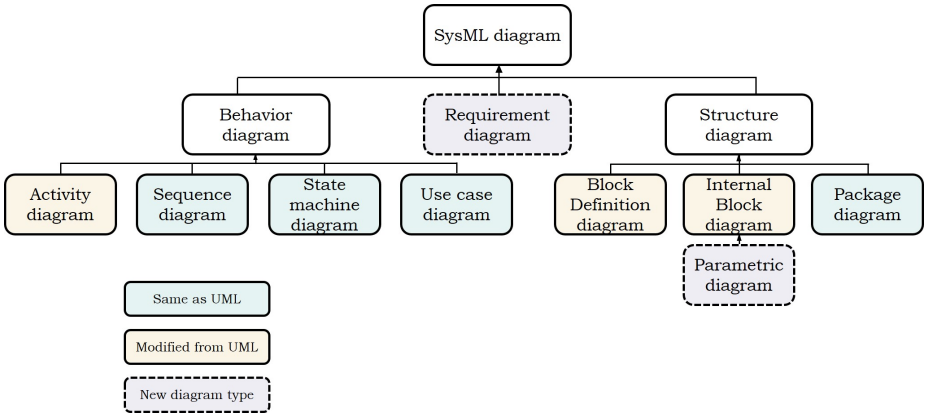


FIGURE 1.8 – Les diagrammes SysML [43]

1.4 Gestion des connaissances dans la conception collaborative

La gestion des connaissances est un domaine qui ne cesse de devenir de plus en plus répandu au sein des industries qui cherchent à sauvegarder leur savoir-faire ainsi que leur capital intellectuel [44]. Comme on va s'intéresser au partage des connaissances dans la conception collaborative, il convient de souligner les travaux de recherche effectués dans ce contexte.

1.4.1 Solutions pour la gestion des connaissances : État de l'art

Le défi principal pour la conception collaborative des systèmes mécatroniques est généralement lié à la façon dont la connaissance peut être partagée entre les domaines d'expertise utilisés durant le processus de conception [3]. En ce sens, des efforts particuliers ont été menés pour proposer des supports de collaboration et de partage des connaissances. Nous proposons de classifier ces supports en trois classes principales : La première classe revient à présenter les solutions basées sur le langage SysML afin de supporter la collaboration. La deuxième classe met en évidence les solutions basées sur les ontologies et la dernière classe illustre les solutions ayant recours à l'exploitation des paramètres et des contraintes durant le processus de collaboration pour supporter l'échange entre les différentes parties prenantes impliquées dans la collaboration.

Solutions basées sur le langage SysML

En ayant recours au langage SysML, des approches ont été proposées dans la littérature pour supporter le partage des connaissances durant la conception collaborative. Un modèle de vue architecturale nommé 3+SysML a été proposé afin de traiter l'intégration synergique des différentes disciplines impliquées dans le développement des systèmes mécatroniques [45]. Le langage SysML est utilisé dans cette approche pour spécifier le modèle de vue centrale du système mécatronique, cependant les trois autres vues concernent les disciplines de la mécanique, l'électronique et l'informatique (Voir figure 1.9).

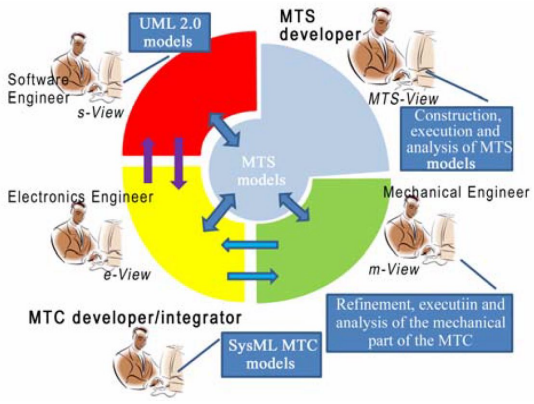


FIGURE 1.9 – Le modèle de vue "3+1 SysML" pour le développement des systèmes mécatroniques [45]

Un profil SysML a été défini à l'aide de l'outil Papyrus [1]. Ce profil supporte la modélisation des systèmes mécatroniques et facilite aussi la réutilisation des connaissances.

Dans le même contexte, une plateforme désignée par SLIM acronyme de *Systems Lifecycle Management* a été développée en se basant sur le langage SysML afin de fournir des solutions assurant un échange et une collaboration entre les modèles SysML et les modèles de l'ingénierie disciplinaire [46]. Cette plateforme permet aux ingénieurs, dès les premières phases de développement du système mécatronique, la vérification automatisée des exigences, la simulation du système et les révisions de conception (figure 1.10). SLIM fournit un environnement de conception homogène pour le développement des systèmes complexes en tenant en compte des organisations et d'environnement. Cependant, la traçabilité de la collaboration ainsi que la résolution des conflits ne sont pas prises en compte dans cette plateforme.

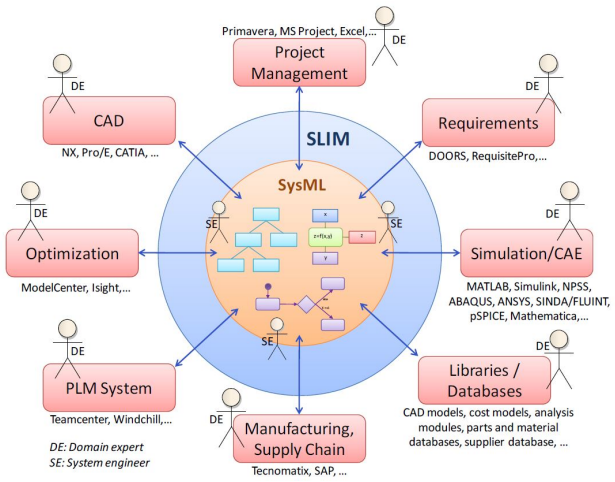


FIGURE 1.10 – Architecture du modèle SLIM [46]

Une autre méthodologie basée sur le langage SysML a été proposée pour l'évaluation intégrée de la conception mécatronique appelée SysDICE (figure 1.11). Le langage SysML est utilisé dans cette approche comme un langage commun entre les ingénieurs disciplinaires qui ont des informations concrètes sur le système et les ingénieurs système qui possèdent une vision globale du système. Dans cette méthodologie, les ingénieurs systèmes et les ingénieurs disciplinaires génèrent les modèles du système, fournissent une formulation mathématique liée à ces modèles et les évaluent [47]. Ces trois étapes sont générées jusqu'à ce que l'ingénieur système obtienne la solution optimale requise. Cette proposition assure et facilite la collaboration entre les ingénieurs disciplinaires et les ingénieurs système mais ne propose pas des solutions pour la réutilisation des résultats de collaboration.

1. Eclipse, `https://www.eclipse.org/papyrus/`

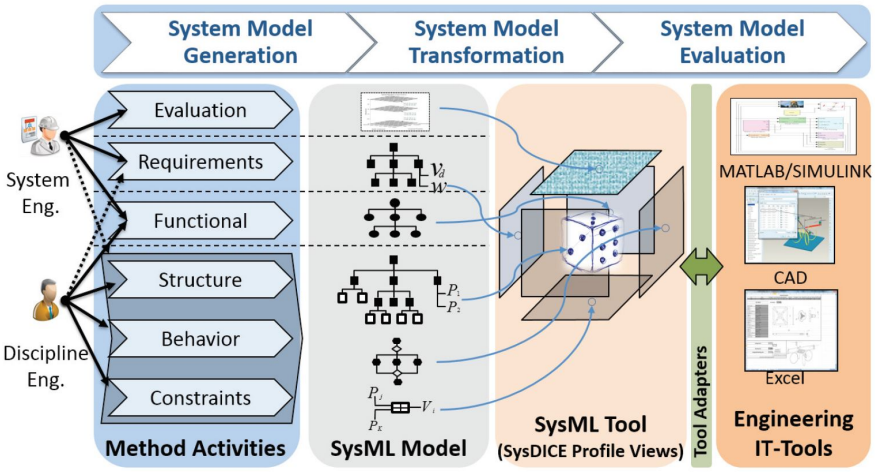


FIGURE 1.11 – Architecture de la plateforme SysDICE [47]

Les méthodologies présentées dans cette section ont démontré leur capacité à simplifier considérablement la tâche de l'ingénieur système, particulièrement pour la vérification de l'architecture du système dès les premières phases. Cependant, la traçabilité de la collaboration et l'échange dynamique des connaissances ne sont pas pris en compte. Pour garantir alors la traçabilité dans le processus collaboratif, des solutions basées sur les ontologies ont été développées. La section suivante décrira ces solutions.

Solutions basées sur les ontologies

Les ontologies représentent un système de concepts fondamentaux qui permet au concepteur de représenter et de modéliser un domaine particulier. Cette approche est utilisée dans les domaines de la gestion des connaissances afin de résoudre des problèmes sémantiques et faciliter le partage des connaissances [48]. Les ontologies ont été employées dans divers domaines pour la représentation et la structuration des connaissances. Nous présentons dans cette section quelques travaux basés sur les ontologies.

Tudorache [48] a utilisé les ontologies comme étant une méthode pour améliorer la qualité des modèles disciplinaires et de soutenir les parties prenantes dans le processus de collaboration. Dans cette approche, les ontologies ont été utilisées afin de vérifier la cohérence entre les modèles hétérogènes et propager les changements d'un modèle à un autre en utilisant un graphe d'ontologie. Ces ontologies ont été employées dans le processus de gestion des exigences et ont soutenu la tâche d'affinement itératif d'une spécification jusqu'à l'obtention d'une conception optimale. Cependant, la collaboration entre les ingénieurs disciplinaires et les ingénieurs

système n'est pas tenue en compte. Dans la même direction, Bi et al. [49] ont proposé une méthode basée sur les ontologies pour définir et décrire les informations communes d'un système mécatronique. Les auteurs ont spécifié une ontologie du système mécatronique nommée *Mechatronic System Ontology* (MOS). Cette ontologie contient tous les objets principaux du processus de conception ainsi que leurs relations. Dans cette approche, l'ontologie du système global est divisée en d'autres sous-ontologies vu la nature multidisciplinaire des systèmes mécatroniques (voir figure 1.12). Grâce à cette approche, une représentation commune des connaissances est fournie ce qui facilite le partage ainsi que la réutilisation des connaissances. Cependant, les contraintes du système mécatronique doivent être prises en considération dans l'ontologie du système (MOS).

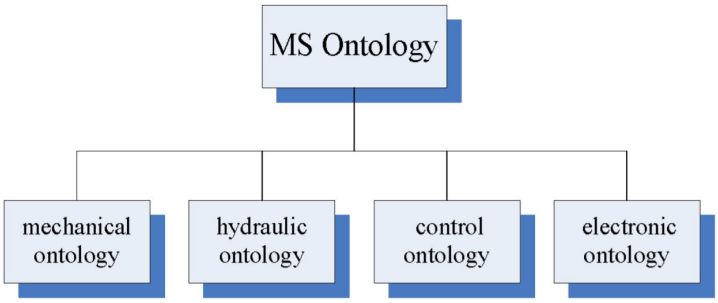


FIGURE 1.12 – Architecture de l'ontologie du système mécatronique (MOS) proposée dans [49]

Une autre approche a été proposée par Hong et al. afin d'établir un modèle de gestion des connaissances approprié pour soutenir l'intégration et le partage des connaissances [50]. Ce modèle est basé sur les ontologies pour obtenir une représentation non ambiguë de la connaissance. Un cadre de conception intégrée homme-machine basé sur l'ontologie est décrit en premier lieu. Ensuite, les ontologies et sous-ontologies correspondantes sont établies selon différents objectifs. Une méthode d'intégration d'ontologies basée sur le calcul de similarité est introduite. Enfin, une approche de partage des connaissances basée sur l'ontologie est développée. En ce sens, une autre proposition a été développée par Wang et al. [51] en se basant sur l'ontologie pour les systèmes mécatroniques complexes. Cette approche permet de soutenir l'interopérabilité sémantique entre les systèmes hétérogènes utilisés dans la conception collaborative. Ceci se fait suivant quatre étapes. La première étape consiste à préparer les documents de conception du produit et la deuxième concerne l'extraction des concepts clés de conception. Ces deux étapes sont suivies de la correspondance des concepts. Finalement, les concepts et leurs relations sont reconnus et définis. Cette approche permet la gestion des connaissances mais ne tient pas en compte la prise de décision dans le processus de conception.

Dans le même contexte et en se basant sur le concept d'ontologie, une approche

permettant de formaliser des points de vue hétérogènes tels que l'ingénierie système, la sûreté de fonctionnement, la topologie, le domaine multi-physique, etc. est établie par Plateaux et al. [52]. Cette ontologie a été développée afin de répondre aux objectifs de cohérence et d'efficacité (Voir figure 1.13). Cependant, cette ontologie n'a pas été mise en œuvre.

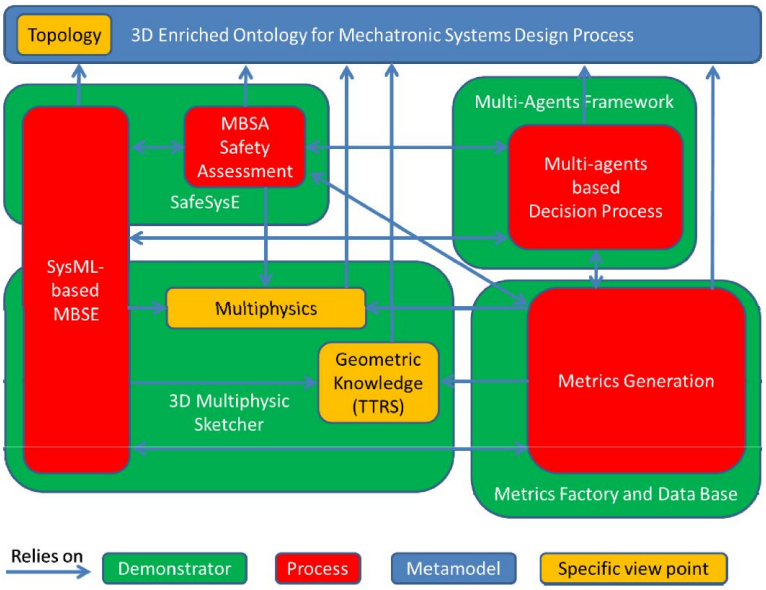


FIGURE 1.13 – Structure de l'approche proposée dans [52]

Les approches citées précédemment permettent de représenter et décrire les connaissances d'un système mécatronique d'une manière claire et non ambiguë. Elles permettent aussi de vérifier la cohérence et d'avoir une vue globale du système mais elles ne proposent pas des moyens d'échange dynamique entre les différentes parties prenantes impliquées dans la conception mécatronique. En ce sens, diverses approches basées sur les modèles paramétriques ont été proposées pour garantir un échange dynamique entre les différentes disciplines. Nous présentons quelques exemples dans ce qui suit.

Solutions pour la gestion des connaissances au niveau des paramètres et des contraintes

Durant le processus de conception mécatronique, divers paramètres peuvent apparaître et doivent être échangés entre les différentes parties prenantes pour assurer une collaboration efficace. De ce fait, plusieurs solutions basées sur la gestion des paramètres et des contraintes (ou règles métiers) ont été proposées dans la littérature. Nous décrivons dans cette section les approches les plus pertinentes dans ce

contexte.

Un outil pour organiser et répertorier le savoir-faire des parties prenantes dans un projet appelé *Engineering Entreprise Knowledge System* (E2KS) a été proposé [53]. Cette approche permet de capitaliser les savoir-faire des concepteurs sous forme de paramètres, règles, graphiques, etc. et de les réutiliser dans des check-listes correspondantes à des vues métiers particulières (Figure 1.14). Ces check-listes peuvent être synchronisées avec les vues métiers. Cependant, cette approche ne permet pas de gérer les conflits entre les utilisateurs [44]. Les connaissances cruciales capitalisées dans ce modèle ne sont pas extraites d'une manière formelle ce qui peut être chronophage.

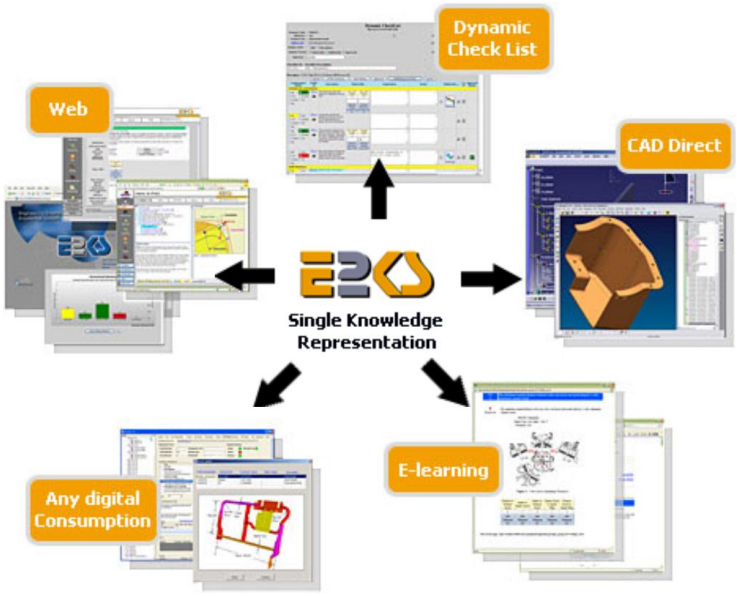


FIGURE 1.14 – Structure de l'approche *Engineering Entreprise Knowledge System* [53]

Dans le même contexte, Kleiner et al. [54] ont proposé un modèle paramétrique, basé sur l'échange et la capitalisation des paramètres de granularité fine (c'est-à-dire des paramètres et des règles métiers), afin d'intégrer les données du produit. Ce modèle est expérimenté à travers l'application COLIBRI (*COnstraint LInking BRIdge*). Il permet de mettre en relation les paramètres encapsulés dans les modèles métiers hétérogènes comme nous pouvons le constater par la figure (1.15). L'emploi de mécanisme de vérification et de résolution permet de tenir en considération différentes contraintes selon différents points de vue. Grâce à cette proposition, il est possible d'intégrer des modèles métiers en utilisant les contraintes. Néanmoins, ce modèle est assez limité en termes de réutilisation, capitalisation et organisation des paramètres et des contraintes. De plus, la gestion des conflits n'a pas été abordée dans cette approche.

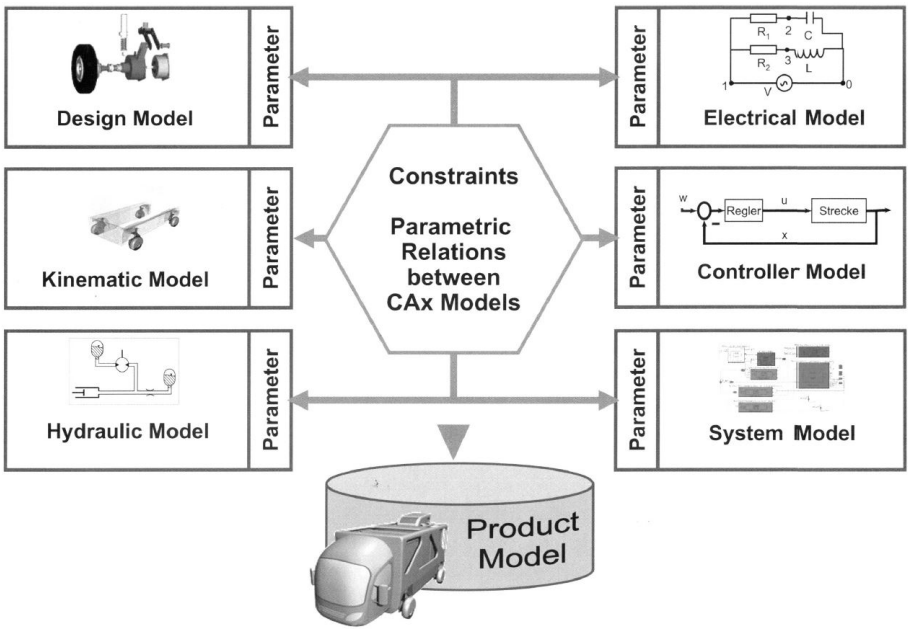


FIGURE 1.15 – Concept de l'approche COLIBRI [54]

Inspiré par le modèle COLIBRI, une autre méthodologie désignée par KCMethod (*Knowledge Configuration Method*) a été proposée. Cette méthodologie s'articule autour la gestion des paramètres et des contraintes qui sont encapsulés dans les modèles métiers de conception et de simulation dans le contexte de l'ingénierie collaborative [10]. Cette proposition se base sur trois concepts fondamentaux comme illustré dans la figure (1.16). L'ICE, abréviation de «*Information Core Entity*», est le premier concept. C'est une entité générique indécomposable qui permet de capitaliser les données cruciales extraites de modèles métiers afin de passer de l'état de donnée vers l'état d'information. Elle contient des paramètres et des règles. La configuration des connaissances ou *Knowledge Configuration* (KC) est le deuxième concept de KCMethod. Cette configuration contient les ICEs qui vont être utilisées durant la collaboration. Au cours de l'étape initiale du projet, les paramètres ne peuvent pas avoir des valeurs. Le dernier concept est la configuration utilisateur ou *User Configuration* (UC) qui contient les instances des ICEs. Elle représente l'interface d'expert et elle est synchronisée avec les modèles métiers. A ce niveau, des valeurs peuvent être attribuées à chaque paramètre. Ce modèle assure la capitalisation, traçabilité, cohérence et réutilisation des connaissances encapsulées dans les différents modèles métiers de conception. Ce modèle a été appliqué dans divers travaux antérieurs [55, 56, 57, 58]. Cependant, KCMethod semble limité en termes d'aide à la décision et d'extraction formelle des connaissances cruciales. De plus, en instanciant l'ICE globale, les utilisateurs peuvent trouver des paramètres inutiles pour leurs activités de conception.

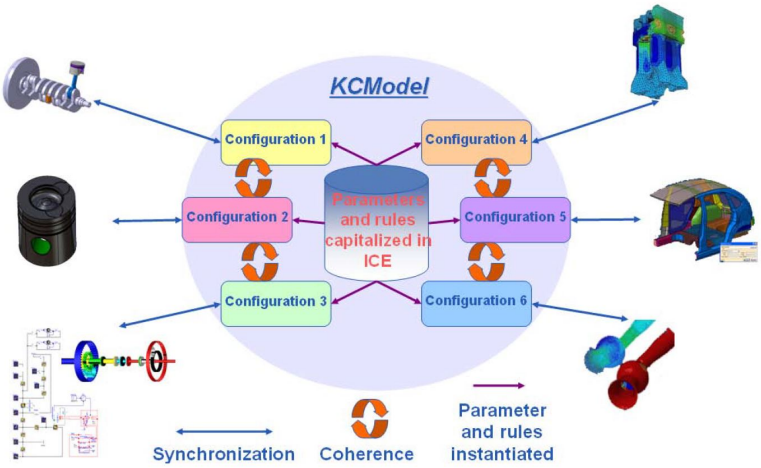


FIGURE 1.16 – Architecture de l'approche KCMethod [10]

Pour remédier, alors, à ces limites, Mcharek et al. [7] ont proposé un nouveau modèle appelé « *Collaborative Design Process and Product Knowledge*» (CDPPK). L'objectif fondamental de ce modèle est d'organiser les paramètres afin de faciliter la collaboration. Le concept d'ICE du modèle précédent a été réutilisé dans ce contexte comme étant une structure conceptuelle formalisée pour assurer une connexion efficace entre les ingénieurs système et les ingénieurs disciplinaires. Pour chaque paramètre un flux d'entrée/sortie est identifié afin d'assurer une connexion entre les ingénieurs disciplinaires. A côté des ICEs de nouveaux concepts ont été proposés comme détaillé dans la figure (1.17), tels que le « *Design Process Knowledge* » (DPK) qui contient toutes les ICEs utilisées dans le projet et la configuration appelée « *User Process Configuration*» (UPC) qui contient les instances d'ICEs.

Un autre concept du modèle CDPPK est le «*Collaborative Design Process*» (CDP). Ce concept englobe toutes les UPCs du projet et permet de gérer les conflits entre les ingénieurs disciplinaires. Le modèle CDPPK a été appliqué sur un système mécatronique dans le domaine d'automobile et a montré son efficacité en termes de partage des connaissances entre les ingénieurs disciplinaires et les ingénieurs système ainsi qu'en termes de gestion des conflits. La réutilisation des connaissances dans d'autres projets ultérieurs est aussi prise en compte dans cette proposition. Cependant, ce modèle ne présente pas des méthodes formelles pour l'extraction des connaissances cruciales. De plus, une seule architecture du système a été prise en compte, alors que les systèmes mécatroniques peuvent avoir diverses solutions ou architectures. Par conséquent, l'aide à la décision dans le modèle CDPPK n'a pas été aussi tenue en compte ce qui peut provoquer un mauvais choix d'architectures.

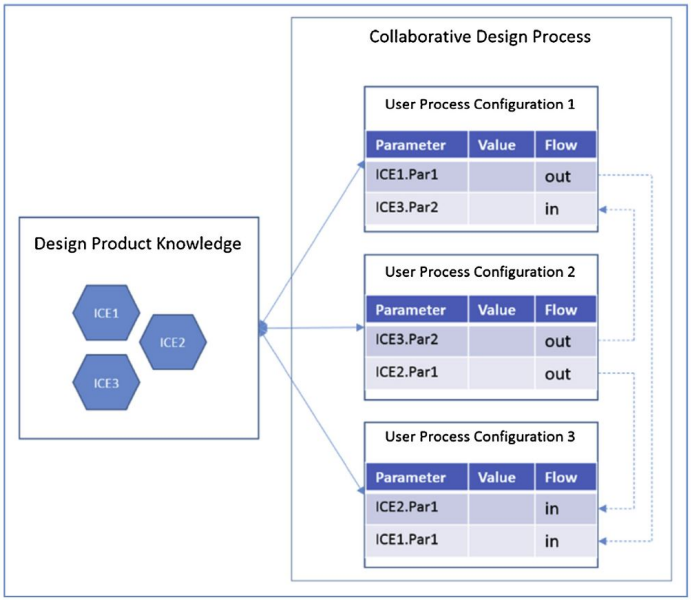


FIGURE 1.17 – Architecture du modèle CDPPK [7]

1.4.2 Synthèse bibliographique

En se basant sur l'étude bibliographique décrite dans la section précédente, nous avons identifié quelques lacunes dans les solutions proposées pour supporter la conception collaborative des systèmes mécatroniques. Nous présentons les efforts que nous allons mener dans notre travail de recherche pour remédier à ces limites. Le tableau (1.1) illustre cette synthèse bibliographique. En s'appuyant sur l'analyse bibliographique, nous constatons qu'un environnement de collaboration supportant l'échange dynamique entre les différents participants dans la collaboration est nécessaire. Cet environnement doit également tenir en compte l'extraction formelle des connaissances cruciales afin de faciliter leur capitalisation. Il a été remarqué qu'au niveau des solutions proposées, la gestion des conflits n'a pas été explicitement présentée. De ce fait, une méthode pour localiser et gérer les conflits dans l'environnement de collaboration devient nécessaire. De plus, l'évaluation de plusieurs alternatives du système n'a pas été traitée.

Notre objectif consiste à proposer un environnement de collaboration pour le développement des systèmes mécatroniques. Dans cet environnement un échange des connaissances de granularité fine (c'est-à-dire des paramètres et contraintes) entre les différents modèles métiers doit avoir lieu.

Pour aboutir à cet objectif, les modèles paramétriques semblent intéressants. Ce choix est justifié par la nécessité de partager des connaissances de granularité fine pour pouvoir aider le chef de projet dans son choix d'architecture. Par consé-

quent, une comparaison entre les solutions basées sur les paramètres et contraintes, décrites précédemment, est établie pour positionner notre travail de recherche. La comparaison s'appuie sur les critères suivants :

— Identification formelle des connaissances cruciales : consiste à extraire les connaissances dont leur contribution à atteindre les objectifs d'un projet est importante.

TABLE 1.1 – Les limites des travaux antérieurs et les objectifs à atteindre

Limites des travaux antérieurs	**Objectifs à atteindre**
Dans les modèles basés sur SysML la traçabilité de la collaboration n'a pas été prise en compte ce qui rend l'extraction des connaissances cruciales et la gestion des conflits difficiles.	On vise dans notre travail de recherche à proposer une démarche pour extraire les connaissances d'une façon formelle ce qui facilitera la traçabilité de ces connaissances ainsi que leur réutilisation.
Dans les solutions basées sur les ontologies l'échange dynamique entre les différentes disciplines impliquées dans la collaboration n'a pas été traité. Des vues holistiques du système ont été proposées sans assurer un échange dynamique dans un environnement commun entre les acteurs du projet.	Afin d'assurer l'échange dynamique entre les différents domaines, on va proposer un modèle de collaboration qui prend en compte l'extraction formelle des connaissances cruciales et qui assure un échange dynamique.
Dans les modèles basés sur SysML et les ontologies, les données de granularité fine ont été ignorées ce qui rend la comparaison entre les différentes architectures difficiles.	En se basant sur les modèles paramétriques, on proposera un modèle qui prend en compte les données de granularité fine tout en assurant une extraction formelle des données et un échange dynamique.
Les modèles paramétriques concrétisent les données de granularité fine mais ne prennent pas en compte que le système peut avoir différentes architectures qui nécessitent la comparaison.	On proposera en ce sens, un modèle de collaboration qui supporte l'extraction formelles des connaissances cruciales tout en permettant l'échange dynamique et la comparaison entre différentes architectures.

— Capitalisation des connaissances : représente le stockage des connaissances cruciales dans des bases de données ou des entités formelles et génériques ce qui facilite leur partage entre les parties prenantes d'un projet.

— Gestion des conflits : permet de comparer les paramètres utilisés dans différents modèles métiers afin de vérifier leur compatibilité. Le respect des contraintes ou des règles métiers durant la collaboration est aussi vérifié à ce niveau.
— Évaluation et choix d'architectures : consiste à observer, interpréter et analyser les résultats de la collaboration afin d'orienter le chef de projet vers l'architecture ou l'alternative la plus adéquate.

A l'aide de ces critères, une étude comparative entre les modèles basés sur les paramètres et contraintes (Modèles E2KS, COLIBRI, KCMethod et CDPPK) est résumée dans le tableau (1.2). Cette étude nous a permis de voir de plus près la position de notre travail de recherche par rapport aux travaux existants dans la littérature.

Dans le modèle E2KS, l'extraction des connaissances cruciales n'a pas été traitée. Ces connaissances sont stockées dans des entités appelées « *Knowledge Packet* » (KPac). Elles permettent de réutiliser les connaissances dans des check-listes correspondantes à des vues métiers particulières. Dans cette approche, la comparaison entre les différentes check-listes pour détecter les conflits entre les utilisateurs n'est pas possible. De plus, l'aide à la décision et l'évaluation de différentes architectures ne sont pas prises en compte.

Par ailleurs, l'approche COLIBRI ne présente pas des méthodes explicites pour extraire les connaissances cruciales. Ce modèle permet de mettre en relation les paramètres issus des modèles métiers via l'intégration de contraintes, en prenant compte la structure du produit et les documents du projet. Cependant, COLIBRI reste limité en termes de capitalisation des connaissances. La gestion des conflits et le choix d'architectures n'ont pas été traitées dans cette approche.

Comme l'approche KCMethod est inspirée du modèle COLIBRI, des efforts ont été menées afin de capitaliser les connaissances cruciales sous forme d'entité d'information (ICE). Les concepteurs instancient les ICEs dont il ont besoin dans les configurations d'utilisateur correspondantes comme nous l'avons déjà souligné dans la section précédente. Les conflits sont détectés dans cette approche si des paramètres ne respectent pas les contraintes ou une valeur différente est allouée au même paramètre dans deux configurations mais une solution pour résoudre les conflits localisés n'a pas été détaillée dans KCMethod. De plus, l'identification des connaissances cruciales et l'évaluation de différentes alternatives du système n'ont pas été prises en compte.

En se basant sur le même principe que l'approche KCMethod, le modèle CDPPK a été développé. Une attention particulière a été accordée à la capitalisation des connaissances cruciales où une ICE décomposable a été proposée comme solution. Grâce à cette décomposition, les concepteurs ont la flexibilité d'instancier seulement les paramètres dont il ont besoin sans instancier l'ICE globale. Cependant, l'identifi-

cation des connaissances cruciales se fait en se basant sur la proposition des réunions avec tous les membres du projet afin de se mettre d'accord sur le choix des connaissances à capitaliser. Cette méthode est difficile à organiser dans un environnement distribué et collaboratif. Dans le modèle CDPPK, les conflits détectés entre plusieurs modèles métiers sont répertoriés dans le *Collaborative Design Process*. Néanmoins, ce modèle reste limité en termes de résolution des conflits et évaluation d'architecture.

TABLE 1.2 – Étude comparative des modèles basés sur les paramètres et les contraintes

Critères	Identification des connaissances	Capitalisation des connaissances	Gestion des conflits	Choix d'architecture
E2KS	Non	KPac	Non	Non
COLIBRI	Non	Contraintes	Non	Non
KCMethod	Manuelle	ICE	Partielle	Non
CDPPK	Manuelle	ICE décomposable	Partielle	Non

Cette étude comparative a abouti au positionnement de notre travail de recherche par rapport aux travaux issus de la littérature. De notre point de vue, CDPPK semble être la solution la plus aboutie avec notre problématique. Cependant, ce modèle ne prend pas en considération la variété d'architectures pour le même système. En effet, les systèmes mécatroniques sont multidisciplinaires ce qui provoque un nombre élevé d'architectures. Ceci implique l'emploi des méthodes d'aide à la décision afin de guider et orienter le chef de projet vers la solution qui satisfait au mieux les exigences. Cette tâche n'est pas prise en compte dans CDPPK. De plus, l'identification des connaissances cruciales en proposant des réunions et des échanges étroits entre les différents acteurs du projet peut être une étape chronophage ce qui représente un inconvénient pour la conception collaborative mécatronique. Ceci est également difficile à organiser surtout avec des environnements complexes et distribués. En ce sens, une méthode formelle pour identifier ces connaissances devient nécessaire. Par ailleurs, les conflits sont seulement détectés dans ce modèle sans proposer des méthodes explicites pour les résoudre.

Dans ce contexte, notre objectif consiste à proposer un environnement de collaboration qui tient en compte, à la fois, l'identification des connaissances cruciales, la gestion des conflits et l'évaluation d'architectures. Le modèle CDPPK semble être intéressant dans notre contexte grâce à ses concepts flexibles pour capitaliser les connaissances cruciales et harmoniser le processus de conception collaborative. Par

conséquent, il sera considéré comme un point de départ pour notre méthodologie. Néanmoins, ce modèle présente certaines verrous d'identification des connaissances cruciales, de résolution des conflits et d'évaluation d'architectures qui doivent être levés. Pour cette raison, il s'avère utile de dresser un état de l'art sur l'identification des connaissances cruciales pour les systèmes complexes. Une étude bibliographique sur les méthodes de gestion des conflits et d'aide à la décision sera également menée dans ce qui suit.

1.5 Identification des connaissances cruciales pour la conception collaborative

Les connaissances cruciales peuvent être définies comme étant des connaissances essentielles et nécessaires pour concevoir un système et pour résoudre les problèmes portant sur un objectif donné, et qui devraient être capitalisées [59]. Saad et al. [9] ont considéré que les connaissances cruciales sont celles dont la contribution pour atteindre les objectifs du projet est très importante et le risque de leur perte et le coût de leur re-création sont considérés comme importants. Uniquement ces connaissances nécessitent la capitalisation. Ce concept sera détaillé dans cette section et les différents travaux de recherche effectués dans le contexte d'identification des connaissances cruciales seront aussi développés.

1.5.1 Capitalisation des connaissances

Selon Grundstein [60], la capitalisation des connaissances de l'entreprise consiste à « considérer les connaissances utilisées et produites par l'entreprise comme un ensemble de richesse ». Dans ce contexte, Grundstein a proposé un cycle de capitalisation des connaissances qui s'articule autour de cinq facettes comme présenté dans la figure (1.18) :

- Repérer : dans cette première étape, les connaissances explicites et les savoir-faire sont identifiés, localisés et caractérisés. L'estimation de la valeur économique de ces connaissances est établie dans cette étape.
- Préserver : une fois toutes les connaissances sont repérées, elles doivent être modélisées, formalisées ainsi que conservées.
- Valoriser : la valorisation des connaissances consiste à les rendre accessible en tenant compte de certaines règles de sécurité. A ce niveau, les connaissances de l'entreprise sont diffusées et partagées. Des nouvelles connaissances peuvent être créées dans cette phase.
- Actualiser : La quatrième facette revient à évaluer, mettre à jour et enrichir les connaissances, en se référant aux retours d'expériences.

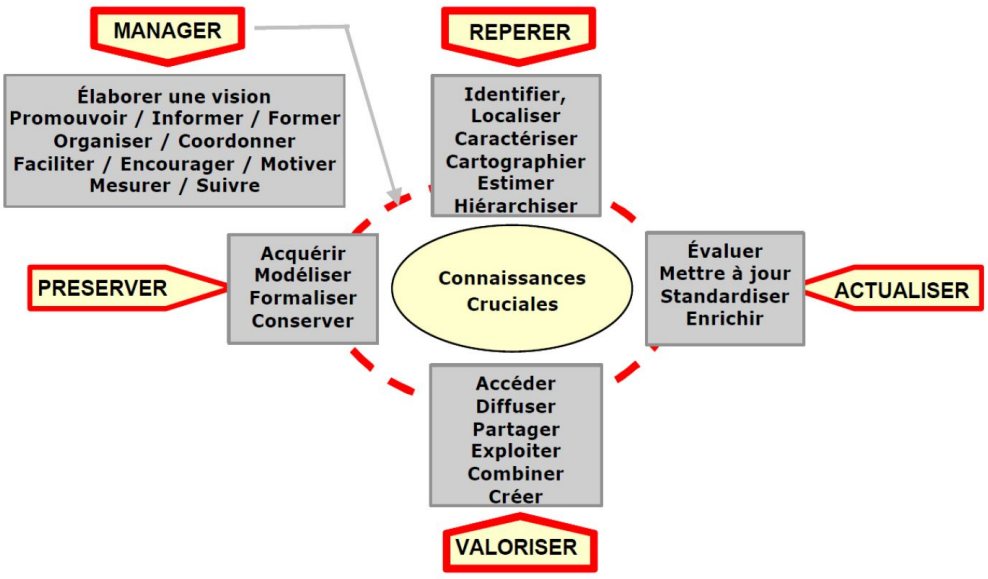


FIGURE 1.18 – La capitalisation des connaissances selon Grundstein [60]

— Manager : cette phase concerne les interactions entre les différentes facettes détaillées précédemment. A ce niveau, le concept de management des activités et des processus se positionne.

Comme nous l'avons indiqué précédemment, seules les connaissances cruciales nécessitent la capitalisation. Dans ce contexte différents travaux de recherche ont été effectués pour identifier les connaissances utiles à la collaboration. Un état de l'art de ces travaux sera dressé dans la section suivante.

1.5.2 Solutions pour l'identification des connaissances cruciales

L'identification des connaissances cruciales a fait l'objet de nombreuses recherches scientifiques. Grundstein a proposé un cadre désignée par GAMETH, acronyme de *Global Analysis METHodology* afin d'identifier les connaissances cruciales à capitaliser [61]. Le cadre GAMETH est constitué de trois étapes (voir figure 1.19). La première étape fournit la spécification de l'espace du problème et du contexte opérationnel. La deuxième étape vise à localiser et à caractériser les connaissances cruciales. La dernière étape revient à évaluer la valeur des connaissances cruciales et déterminer les objectifs de gestion des connaissances. Dans cette approche, l'analyse des connaissances est basée sur les expériences du chef de projet.

En outre, Pomian et Roche [62], ont constaté que les connaissances cruciales peuvent être identifiées par des entretiens avec le chef de projet. De la même ma-

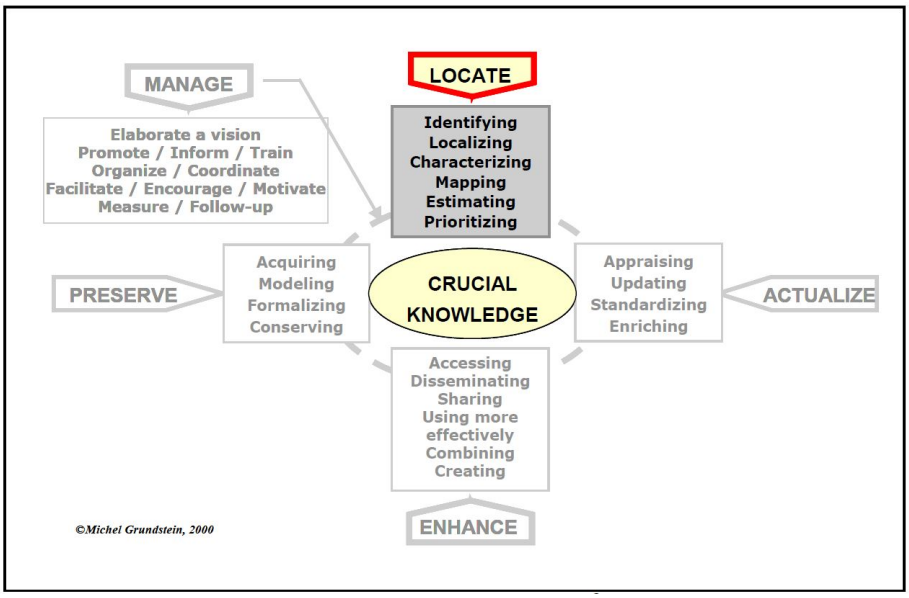


FIGURE 1.19 – Positionnement du cadre GAMETH par rapport à l'approche de la capitalisation des connaissances [61]

nière, Ermine [63] a proposé une série d'entretiens avec le chef de projet ainsi que des études de documents stratégiques afin d'identifier les connaissances les plus précieuses. Le même concept a été utilisé par Ammar-Khodja et al. dans leur travail de recherche [64]. Les auteurs ont proposé un processus d'ingénierie des connaissances afin de structurer ces connaissances. Ce processus est basé sur la méthodologie MOKA, un acronyme qui fait référence à *Methodology and tools Oriented to Knowledge-based engineering Applications*. L'approche proposée permet de localiser les connaissances cruciales par le biais d'entretiens et d'une communication étroite entre toutes les équipes impliquées dans la conception. En se basant sur le même principe, Badin et al. [10] et Mcharek et al. [7] ont recommandé une réunion avec les différentes parties prenantes pour identifier les connaissances nécessaires dans les modèles paramétriques. Ces approches sont basées sur l'intuition du décideur, ce qui peut influencer leur fiabilité. En outre, les réunions avec tous les membres du projet peuvent être difficiles à organiser dans un environnement distribué et nécessitent un temps d'exécution considérable.

Par ailleurs, des approches intéressantes basées sur des méthodes d'aide à la décision ont été proposées. Tseng et Huang [59] ont proposé de calculer le score de chaque connaissance et d'évaluer l'importance de ces connaissances. De plus, Saad et al. ont proposé d'étendre le cadre GAMETH [9]. L'approche se compose de deux étapes principales. Le développement d'un modèle de préférences des décideurs est présenté dans la première étape. Ce modèle vise à déterminer les règles de décision en fonction de préférence des décideurs. La deuxième étape de l'approche proposée

revient à classifier les connaissances. Une méthode d'aide à la décision multicritère est utilisée dans cette recherche afin de prendre en considération les préférences du décideur. Ce travail de recherche a été étendu par Brigui-Chtioui et Saad [65] et Ghrab et al. [66] où des nouvelles classifications des connaissances ont été présentées pour choisir les connaissances cruciales tout en utilisant des méthodes d'aide à la décision. De même, Hassan et al. ont développé un cadre d'évaluation multicritère qui supporte l'identification et la localisation des connaissances cruciales [67]. Les auteurs font appel aux méthodes de décision multicritère afin d'améliorer la localisation des connaissances cruciales créées et mobilisées par les modèles métiers. Néanmoins, ces approches nécessitent un temps important de mise en œuvre et un grand nombre de participants, ce qui peut compliquer davantage la collaboration.

1.5.3 Synthèse bibliographique

De nombreux efforts ont été menés dans la littérature pour identifier les connaissances cruciales. Ces travaux de recherche sont basés soit sur l'organisation des réunions avec tous les participants impliqués dans le processus de conception, soit sur l'intégration de certaines méthodes ADMC. Comme nous l'avons mentionné précédemment, l'identification des connaissances cruciales basée sur l'organisation des réunions avec tous les acteurs du projet peut être difficile à organiser, particulièrement pour les systèmes mécatroniques à cause de leur complexité. De plus, l'intégration des méthodes d'aide à la décision nécessitent de nombreux intervenants, ce qui rend le processus de conception plus complexe. Par conséquent, la mise en place d'un cadre formel pour extraire les connaissances les plus importantes devient nécessaire. La formalisation des connaissances cruciales apporte, d'une part, une unification des modèles métiers hétérogènes, ce qui convient à la conception collaborative mécatronique où un environnement homogène est nécessaire. D'autre part, cette formalisation des connaissances évite la nécessité d'un échange direct entre les parties prenantes, qui peut être difficile à organiser dans un environnement complexe comme celui de la conception mécatronique, où un grand nombre de connaissances est utilisé. Ainsi, la théorie mathématique appelée "Théorie des catégories" (TC) est considérée comme un outil puissant pour fournir un cadre unifié et formel afin d'illustrer les interconnexions entre les différents modèles métiers utilisés dans la collaboration. Il serait donc intéressant de présenter dans les sections suivantes de ce chapitre, les concepts fondamentaux ainsi que les travaux de recherche en relation avec la TC. Cependant, comme on vise à proposer une méthodologie qui supporte non seulement l'identification des connaissances cruciales, mais également la gestion des conflits et l'évaluation d'architectures pour les systèmes mécatroniques, il s'avère utile, avant d'introduire la TC, de présenter brièvement les approches présentées dans la littérature pour gérer les conflits et évaluer différentes architectures.

1.6 Gestion des conflits et d'incohérences

L'incohérence peut être définie comme un conflit ou une contradiction entre différents éléments du modèle selon Taylor et al. [68]. Cette contradiction peut prendre cinq formes : les incohérences de nom, d'interface, de comportement, d'interaction et de raffinement.

- Une incohérence de nom se produit entre deux éléments indépendants portant le même nom. Un exemple de cette incohérence se présente entre deux composants nommés "capteur", alors que le premier est un capteur de position et le second est un capteur de courant.
- L'incohérence d'interface concerne deux éléments dont les terminologies ou les valeurs ne correspondent pas. Par exemple, cette incohérence peut se présenter lorsqu'une distance "D" a deux valeurs différentes dans deux modèles différents ou lorsqu'une "connexion électrique" est appelée "câblage électrique" dans un autre modèle.
- L'incohérence comportementale se produit lorsque le comportement de deux éléments ne correspond pas. Ce type d'incohérence se produit lorsqu'une distance est exprimée en "millimètres" dans un modèle et en "centimètres" dans un autre.
- L'incohérence d'interaction se produit lorsqu'un élément ne respecte pas certaines contraintes.
- L'incohérence de raffinement se produit lorsque deux modèles de niveaux d'abstraction différents ont des éléments différents afin de correspondre au niveau d'abstraction correspondant. Un exemple d'incohérence de raffinement serait le suivant : dans un modèle, une "unité de commande" est définie comme un composant entier, alors qu'elle est définie comme une combinaison d'un contrôleur, d'un signal de tension et d'un capteur de position dans un modèle plus détaillé.

Les incohérences d'interface et d'interaction sont les types les plus populaires traités par les outils et les travaux de recherche issus de la littérature. Cependant, les incohérences de comportement et de raffinement sont les moins populaires en raison du nombre insuffisant d'outils capturant et gérant ces incohérences selon Torres et al. [13]. Dans notre travail de recherche, nous nous concentrerons sur les incohérences d'interaction et d'interface entre différents modèles issus de différentes disciplines d'ingénierie. Notre hypothèse est de résoudre uniquement les conflits qui se produisent lorsque deux éléments présentent des valeurs incompatibles ou ne respectent pas les contraintes définies dans les exigences du système. Les incohérences de nom, de terminologie, de comportement ou de raffinement n'entrent pas dans le cadre de cette thèse. Les approches de gestion des conflits proposées auparavant seront présentées dans ce qui suit.

1.6.1 Solutions pour la gestion des conflits

Méthodes de gestion des conflits basées sur l'interopérabilité

L'interopérabilité est définie comme la coopération entre deux ou plusieurs composants logiciels, qui sont conçus dans des interfaces, des langages et des plateformes d'exécution différents [69]. Guychard et al. [70] ont fournit une méthodologie pour assurer la cohérence et maintenir la traçabilité basée sur trois espaces interopérables. L'espace d'information contient des bases de données, des fichiers et des outils. Dans l'espace conceptuel, la fédération des modèles est mise en œuvre, alors que l'espace de conception est dédié à la gestion des deux espaces précédents. Cette approche vise à construire une base de données unique où les données peuvent être stockées. Cependant, l'utilisation de cette base de données unique peut présenter certains problèmes de sécurité et implique la gestion d'une grande quantité de données.

Une autre solution est présentée par Qamar et al. [71] où la nécessité de gérer les conflits par l'interopérabilité entre différents outils est discutée. Dans ce travail de recherche, des formats de fichiers standards sont utilisés afin de maintenir l'interopérabilité entre des outils distincts tels que TeamCenter [2], MagicDraw [3] et Simulink [4]. Cette solution assure la cohérence et maintient la traçabilité. Cependant, elle peut présenter des problèmes de perte de données lors de la transmission des données entre des domaines et des outils hétérogènes.

Méthodes de gestion des conflits basées sur les ontologies

En ayant recours aux ontologies, Bock et al. [72] ont proposé une approche dans laquelle les techniques basées sur les modèles et les ontologies sont combinées afin de faciliter la conception collaborative. L'ontologie permet aux différentes parties prenantes de développer des descriptions du produit et de vérifier la cohérence une fois les descriptions combinées. Néanmoins, les auteurs ne présentent pas des solutions pour résoudre et gérer les conflits détectés.

Dans le même contexte, Penas et al. [73] ont supposé que la vérification de la cohérence est une question importante dans les processus de conception des systèmes mécatroniques et des systèmes cyber-physiques ou *Cyber-Physical Systems*(CPS). Les auteurs ont proposé la description des différents sous-systèmes à travers une vue globale basée sur des ontologies. Cette représentation doit être non ambiguë et précise pour garantir l'interprétation correcte des informations stockées et des

2. Automation Siemens, `https://www.plm.automation.siemens.com/global/fr/products/teamcenter/`

3. MagicDraw, Dassault Systèmes, `https://www.3ds.com/products-services/catia/products/no-magic/magicdraw/`

4. Simulink, Mathworks, `https://fr.mathworks.com/products/simulink.html`

décisions des concepteurs. Selon les auteurs, la représentation ontologique peut assurer la cohérence. Cependant, une solution explicite de vérification et de gestion des incohérences n'est pas présentée dans ce travail.

Méthodes de gestion des conflits basées sur la modélisation de dépendance

Divers efforts ont été menés dans la littérature en se basant sur la modélisation explicite de dépendance afin de faciliter la gestion des modèles et la vérification de cohérence. Dans ce contexte, une approche basée sur la modélisation de dépendance pour capturer les inter/intra-dépendances entre les modèles hétérogènes a été proposée [74]. Cette approche vise à informer les participants à la conception des incohérences possibles si les propriétés dépendantes changent. Les auteurs affirment que l'impact du changement d'un modèle peut être prédit par la modélisation de dépendance, ce qui permet de détecter plus facilement les incohérences. Cependant, la résolution de ces incohérences est basée sur un processus manuel par lequel le chef de projet vérifie et gère le conflit détecté. De même, Törngren et al. [27] se sont concentrés sur la modélisation de dépendance entre les modèles, les personnes ainsi que les outils. Dans cette approche, la dépendance est visualisée par une solution de Web sémantique pour les points de vue internes et externes. Le modèle développé peut être utilisé pour signaler les incohérences lorsque des modifications sont apportées aux modèles métiers. Néanmoins, les auteurs ne traitent pas explicitement la résolution de ces conflits.

Méthodes de gestion des conflits basées sur la synchronisation des modèles

L'approche de la synchronisation des modèles est basée sur une transformation unidirectionnelle ou bidirectionnelle entre les modèles impliqués dans le processus de conception. Ainsi, dans cette approche, plusieurs règles de transformation sont formulées pour définir la nature des relations entre les entités des différents modèles. Un exemple de cette approche est le concept de Legendre et al. [75]. Dans cette étude, les auteurs ont présenté une approche collaborative et itérative basée sur la synchronisation des modèles. Ce travail se concentre sur la synchronisation entre l'architecture des modèles de sûreté de fonctionnement et l'évaluation des risques. L'activité de synchronisation est basée sur l'identification des incohérences résultant de la confrontation de deux points de vue. Ces incohérences sont gérées par la recherche de compromis, qui feront évoluer progressivement les différents modèles. Cette proposition est appliquée manuellement à un cas d'étude industriel. Ainsi, pour surmonter cet inconvénient, Berriche et al. [76] ont étendu les travaux précédents vers une méthode automatisée pour formaliser la synchronisation des modèles

dans la conception mécatronique (Voir figure 1.20).

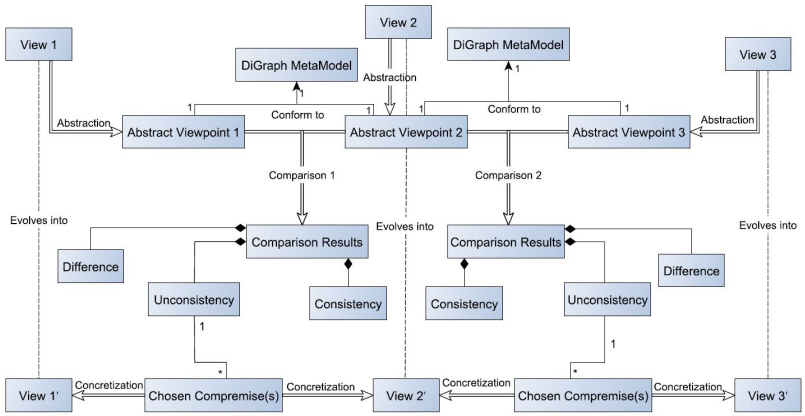


FIGURE 1.20 – Approche de synchronisation des modèles développée par [76]

L'idée principale de cette approche est l'utilisation du standard QVT (*Query View Transformation*) afin de spécifier l'abstraction ainsi que les opérations de concrétisation en utilisant un ensemble de règles de transformation formelles . Cette contribution assure la cohérence entre les modèles spécifiques au domaine de manière automatisée et s'applique aux modèles structuraux et hiérarchiques. Néanmoins, le mécanisme de résolution des conflits est un processus manuel.

La synchronisation des modèles est une solution utile pour maintenir la cohérence entre des modèles hétérogènes. Cependant, cette synchronisation vise à vérifier les cohérences de l'architecture du système et néglige les incohérences des paramètres et des contraintes, qui est l'objectif de notre étude.

Approches de gestion des conflits basées sur les *patterns* et les règles de cohérence

Cette approche est basée sur la création des *patterns* afin de décrire la condition d'existence de toute incohérence. Grâce à ces modèles, le *mapping* des modèles est possible. Ainsi, sur la base de la correspondance des modèles, l'incohérence peut être détectée et traitée en l'ignorant, en la gérant ou en la tolérant [77]. Herzig et al. ont proposé une approche pour automatiser la tâche de gestion des conflits dans le contexte de l'ingénierie système basée sur les modèles (MBSE). Dans cette proposition, les modèles hétérogènes peuvent être représentés par des graphes et les incohérences peuvent être identifiées sous forme de patterns. A ce niveau, les modèles sont transformés en un formalisme de graphe. Des requêtes entre les graphes et les *patterns* qui définissent les différent types d'incohérence sont établies. Feldmann et al. [77] ont relevé le défi de la gestion des incohérences en proposant une approche

globale qui vise à spécifier, détecter et traiter les incohérences dans le MBSE (Voir figure 1.21).

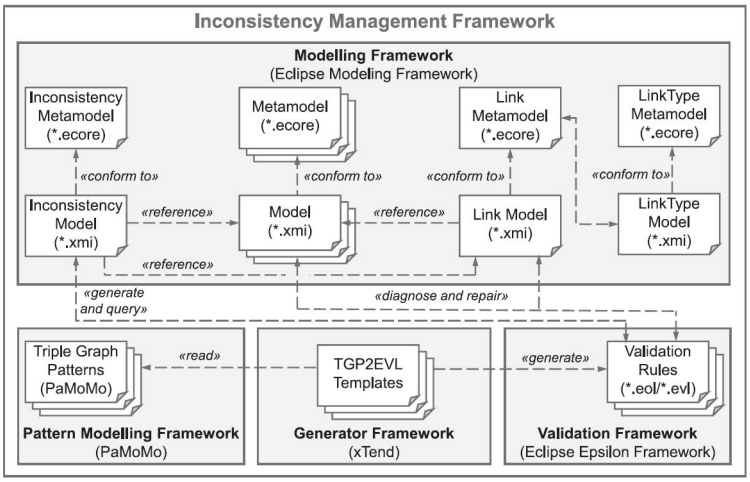


FIGURE 1.21 – Aperçu de l'approche proposée par Feldmann et al. pour la gestion des conflits [77]

Un langage de modélisation graphique dédié est proposé dans cette approche afin de détecter explicitement les conflits et identifier les règles de cohérence qui doivent exister entre les modèles hétérogènes.

Méthodes de gestion des conflits basées sur les données de granularité fine

Cette approche suggère l'utilisation des paramètres et des contraintes pour vérifier la cohérence des modèles hétérogènes au sein d'une équipe pluridisciplinaire. Cette approche se concentre sur les incohérences de valeur et d'interaction selon la classification présentée par Taylor et al. [68]. Une fois que les valeurs d'un paramètre ne sont pas compatibles ou les contraintes ne sont pas respectées, le conflit peut être détecté et doit ainsi être géré pour maintenir la cohérence du système à concevoir. Dans ce contexte, Kleiner et al. [54] ont proposé un modèle permettant d'échanger et de capitaliser des données de granularité fine pour vérifier la cohérence et aider les concepteurs à résoudre les conflits. Néanmoins, cette approche ne présente pas une solution explicite pour gérer les conflits qui ont eu lieu lors de l'intégration des modèles métiers. Par ailleurs, cette approche est étendue par Badin et al. [10] vers le modèle de configuration des connaissances (KCMethod décrit dans la section 1.4.1). Les ICEs sont utilisées par les ingénieurs du projet afin de détecter les conflits potentiels. Cependant, l'approche KCMethod est limitée en termes de résolution des conflits. Des efforts ont été menés par Yvars [78] et Canbaz et al. [21, 79, 80] afin

de résoudre les conflits entre les données de granularité fine utilisées dans des modèles métiers hétérogènes. Dans ces travaux de recherche, le processus de conception est représenté comme étant un problème de satisfaction de contraintes. Une plateforme, centrée sur la notion de contrainte permettant de modéliser et de résoudre un problème de conception, est produite. Cependant, l'unification des modèles métiers hétérogènes n'a pas été proposée dans ces travaux. Inspirés par le modèle KCMethod, Mcharek et al. [7] ont développé un nouveau modèle de collaboration appelé *collaborative design process and product knowledge* (CDPPK) décrit précédemment. De même, ce modèle est limité en terme de gestion des conflits car il propose de les résoudre manuellement en se basant sur les décisions du chef de projet.

1.6.2 Synthèse bibliographique

Une multitude d'approches est proposée dans la littérature dans le contexte de la gestion des conflits. Ces approches sont classées en six types différents. Comme nous l'avons vu précédemment, l'approche basée sur l'interopérabilité présente certains problèmes de perte de données en raison de la transmission de données entre des domaines et des outils hétérogènes. La représentation ontologique assure la vérification de la cohérence pendant le processus de conception, mais la création d'ontologie nécessite un temps d'exécution considérable. En outre, certains efforts dans la littérature basés sur la modélisation explicite de dépendance ont été menés, mais ces solutions ne traitent pas la résolution des conflits. Les approches de synchronisation des modèles sont proposées dans la littérature comme une solution efficace dans le domaine de la gestion d'incohérence. Cependant, cette solution ne prend pas en compte la résolution des conflits entre les données de granularité fine, qui est l'objectif de notre travail de recherche. L'approche basée sur les règles et les patterns a montré son efficacité dans la gestion des conflits de données de granularité fine. Alors que certaines études ne présentent pas des solutions explicites pour gérer les conflits détectés, d'autres travaux de recherche traitent ce problème en proposant plusieurs techniques de résolution des conflits. L'utilisation des règles et patterns tend à être flexible puisque les règles peuvent être supprimées ou ajoutées sans réviser l'ensemble du mécanisme de gestion des conflits. Cela peut être intéressant pour la conception des systèmes mécatroniques, notamment en termes de réduction du temps de développement. Enfin, l'approche basée sur les paramètres et les contraintes est présentée comme une technique de résolution des conflits. Le principe de cette approche est l'idée centrale de notre étude. Néanmoins, les travaux de recherche proposés dans ce contexte ne présentent pas des solutions explicites pour gérer ces conflits.

Par conséquent, pour surmonter cet inconvénient, nous proposerons une nouvelle méthodologie de gestion des conflits entre les données de granularité fine, basée sur des patterns et des règles. Grâce à ces règles, notre proposition sera plus flexible, ce

qui facilitera la tâche de résolution des conflits. Dans les approches basées sur les règles et les patterns mentionnées ci-dessus, des graphes sont utilisés pour créer des patterns et établir des requêtes entre les différents modèles métiers et les patterns. Ces graphes ont montré leur efficacité en rendant le processus de résolution des conflits plus simple. Cependant, l'utilisation de ces graphes peut négliger certaines intuitions issues de la pratique, ce qui est essentiel pour conserver la signification du modèle métier [81]. En outre, la conception mécatronique englobe plusieurs disciplines qui collaborent entre elles, ce qui rend la conception assez complexe. Par conséquent, pour faire face à cette complexité, des méthodes formelles basées sur des techniques mathématiques sont nécessaires [82]. Dans ce contexte, la TC, en tant que théorie mathématique, fournit un cadre d'unification pour formaliser l'interconnexion entre les parties prenantes, ainsi que pour faciliter la localisation et la résolution des conflits. Cette théorie sera détaillée dans les sections suivantes. Avant de détailler les concepts fondamentaux de cette théorie, il est nécessaire de présenter quelques travaux dans le contexte de l'ADMC vu que la comparaison entre différentes architectures d'un système mécatronique est parmi nos objectifs de recherche.

1.7 Aide à la décision multicritère

L'aide à la décision est considérée comme une activité permettant d'obtenir des éléments de réponse à des questions qui se posent par des intervenants dans un processus de décision [83]. Dans la conception mécatronique, différentes alternatives ou architectures d'un système peuvent se présenter. Ainsi, une évaluation de ces alternatives devient nécessaire afin de choisir celle qui répond le mieux aux préférences du décideur. Cette évaluation s'effectue selon des critères multiples, dont certains peuvent être contradictoires [84]. Par conséquent, le recours aux méthodes d'aide à la décision multicritère constitue une solution appropriée pour évaluer différentes alternatives d'un système mécatronique. De ce fait, nous allons mener dans cette section une étude bibliographique permettant d'avoir un aperçu sur les méthodes d'ADMC.

1.7.1 Méthodes d'aide à la décision multicritère (ADMC)

Le vecteur d'indice mécatronique ou encore *Mechatronic Index Vector* (MIV) a été introduit comme une solution pour évaluer plusieurs alternatives dans la conception mécatronique [85]. Cette méthode se compose de trois critères différents à savoir, la flexibilité, la complexité et l'intelligence. Le niveau d'intelligence d'un système est obtenu en se référant aux fonctions de contrôle. Différents types de flexibilité ont été proposés dans cette approche, à savoir, la flexibilité des opérations, la flexibilité de la capacité et la flexibilité du produit.

Par ailleurs, le quotient de conception mécatronique ou en anglais *Mechatronic Design Quotient* (MDQ) a été proposé comme une mesure multicritère pour aider à la prise de décision dans la conception mécatronique [86]. Sept critères ont été incorporés dans le MDQ, notamment la satisfaction des exigences, la fiabilité, l'intelligence, l'adéquation, la facilité de contrôle, l'efficacité et le coût. Ces critères sont agrégés au moyen de l'intégrale de Choquet, une intégrale floue non linéaire qui peut être utilisée pour aider à la prise de décision avec des critères interactifs.

Un nouvel indicateur multicritère appelé *Mechatronic Design Indicator* (MDI) pour l'évaluation de la performance des systèmes mécatroniques a été formulé par Hammadi et al. [87]. Cet indicateur varie de 0 pour une mauvaise performance à 1 pour une bonne performance. Tous les critères de performance sont agrégés dans cet indicateur et les meilleures configurations mécatroniques sont déterminées en maximisant le MDI.

En se basant sur le même principe de MIV et MDQ, le profil multicritère mécatronique ou *Mechatronic Multicriteria Profile* (MMP) repose sur cinq critères, à savoir l'intelligence de la machine, la fiabilité, la flexibilité, le coût de fabrication et de production ainsi que la complexité [88]. Les critères de ce profile sont définis en se référant non seulement à l'expérience du décideur, mais aussi, aux grandeurs mesurables.

En outre, Moulianitis et al. [89] ont proposé un nouvel indice mécatronique pour l'évaluation des alternatives. L'intégrale de Choquet discrète est utilisée pour l'agrégation des scores d'évaluation, tout en prenant en compte les corrélations entre les critères. Ces critères se concentrent sur la capacité d'interaction, de manipulation, mouvement et perception. La capacité cognitive, l'autonomie décisionnelle et la configurabilité sont également considérées comme des critères dans cette approche.

D'autres part, certains travaux de recherche se sont concentrées sur l'emploi des méthodes d'aide à la décision multicritère pour évaluer les différentes alternatives d'un système mécatronique. Dans ce contexte, une méthode d'évaluation d'architectures physiques est proposée par Chen et al. [90]. Dans cette approche, les architectures du système sont évaluées sur la base de la technique de préférence d'ordre par similarité à une solution idéale, désignée en anglais par *Technique for Order Preference by Similarity to Ideal Solution* (TOPSIS). D'autres méthodes d'aide à la décision multicritère sont présentées dans la littérature. Ces méthodes peuvent être classées en deux types : les méthodes du critère unique de synthèse et les méthodes de surclassement.

Approches du critère unique de synthèse

Le principe de cette méthode consiste à maximiser une fonction unique d'agrégation ou d'utilité. Cette approche s'agit d'attribuer des poids pour chaque alternative

afin de déduire directement un classement de toutes les alternatives en comparaison. Parmi les méthodes appartenant à cette classe, on distingue : *Analytical Hierarchical Process* (AHP) [91], *Simple Multi-Attribute Rating Technique* (SMART) [92], *Technique for Order Preference by Similarity to Ideal Solution* (TOPSIS) [93], *Multiple Attribute Utility Theory* (MAUT) [94], etc.

Méthodes de surclassement

L'approche de surclassement de synthèse se base sur la comparaison des alternatives par paires tout en évitant les compensations. Cette méthode génère un classement sous la forme d'un pré-ordre partiel. Deux méthodes basées sur le principe de surclassement se démarquent : la méthode Élimination Et Choix Traduisant la REalité (ELECTRE)[95] et *Preference Ranking Organization METHod for Enrichment Evaluations* (PROMETHEE) [96].

1.7.2 Procédure de choix d'une méthode d'ADMC

Il est important de souligner que les méthodes présentées précédemment se différencient en termes de modélisation de préférence du décideur ainsi que ses attentes. De ce fait, on constate que ces méthodes d'aide à la décision multicritère dépendent des entrées et sorties. Une classification en se basant sur 24 entrées et 7 sorties de ces méthodes a été proposée par Guitouni et al. [97]. Une représentation matricielle, comme nous pouvons le voir sur le tableau (1.3), est fournie pour aider le décideur à choisir la méthode la plus appropriée pour son problème. La définition des sorties (output) et entrées (input) de cette classification est représentée par les tableaux (1.4) et (1.5) respectivement.

Grâce à cette classification, le choix de la méthode d'aide à la décision devient plus facile. Un positionnement dans la case correspondante est suffisant pour trouver la méthode nécessaire à la résolution du problème de décision. En utilisant cette manière, on constate que l'approche AHP est la plus adéquate pour résoudre notre problème de décision. La méthode AHP permet de représenter le problème de décision en se basant sur une structure hiérarchique. Cette structure met en évidence les interactions entre les éléments du problème en premier lieu, et effectue ensuite des comparaisons par paires de ces éléments.

Le choix de la méthode AHP pour notre méthodologie générale est justifié par le fait que cette méthode se base sur un modèle compréhensible et souple pour résoudre les problèmes non structurés. Cette approche tient en compte l'interdépendance des éléments à comparer et ne se concentre pas sur une résolution linéaire. De plus, cette méthode est basée sur l'attribution des valeurs numériques à des jugements subjectifs et sur la synthèse de ces jugements pour déterminer les variables ayant la plus

TABLE 1.3 – Extrait de la classification des méthodes d'aide à la décision selon Guitouni et al. [97]

Input	Output						
	O1	**O2**	**O3**	**O4**	**O5**	**O6**	**O7**
I1							
I2							
I3				ELEC-TRE IV			
I17				ELEC-TRE II	MAC-BETH	ELEC-TRE I	
I18			PROM-THE II	PROM-THE I			ELECT-RE TRI
I19			EVAMIX	ELECT-RE III		ELECT-RE IS	
I20	MAUT/ TOPSIS		MAVT/ SMART				
I21	AHP						

grande priorité. Par ailleurs, notre objectif est la hiérarchisation des alternatives d'un système mécatronique afin d'établir un ordre du degré de gravité de chaque alternative. D'où l'intérêt d'intégrer la méthode AHP dans notre méthodologie générale afin de comparer les différentes alternatives du système mécatronique.

1.8 Problématique et objectifs de la recherche

A l'issue de l'étude bibliographique détaillée précédemment, on constate que les supports de collaboration présentent certaines limites. Dans un premier temps, l'identification des connaissances cruciales s'effectuent en se basant sur des approches manuelles ce qui peut être considérablement chronophage pour choisir les connaissances nécessaires à la collaboration. Dans un second temps, ces supports de collaboration nécessitent une méthode explicite afin de détecter ainsi que gérer les conflits. Finalement, ces supports, présentés dans la littérature, tiennent en compte une seule architecture du système sans proposer des solutions pour comparer différentes architectures du système à concevoir. Ceci nous a conduit à proposer une nouvelle approche de conception collaborative pour les systèmes mécatroniques supportant à la fois, l'identification formelle des connaissances cruciales, la gestion des conflits et la comparaison entre différentes architectures. Notre objectif consiste à

TABLE 1.4 – Caractérisation des sorties proposée par Guitouni et al. [97]

Output	**Signification**
O1	Évaluation globale traduisant la volonté de construire un critère unique de synthèse.
O2	Un rangement global des actions en considérant un seuil d'indifférence qui permet d'introduire des nuances ou limiter certains conclusions
O3	Un rangement total des actions avec d'ex oequo. L'objectif dans ce cas est souvent le rangement des actions de la meilleure à la moins bonne. Dans ce cas, toutes les actions sont comparables : il est possible de les discriminer.
O4	Un rangement partiel des actions en considérant l'incomparabilité. Cet output reprend l'idée de l'output précédent tout en considérant qu'il est possible de ne pas pouvoir discriminer entre certaines actions.
O5	Choix de la meilleure action ou d'un classement d'équivalence de meilleures actions
O6	Choix d'un sous-ensemble d'actions parmi lesquelles ne trouve (nt) la (les) meilleure (s) action (s).
O7	Tri ordonné : affecter les actions à des catégories prédéfinies et ordonnées

développer un support de collaboration pour concevoir des systèmes mécatroniques robustes tout en identifiant d'une manière formelle les connaissances dites cruciales, localisant et gérant les conflits et comparant différentes alternatives du système.

1.9 Questions de recherche

La problématique de notre travail de recherche, présentée dans la section précédente, peut être résumée en quatre questions comme suit. Chaque question correspond à un axe de recherche particulier (voir Figure 1.22).

— Q1 : Comment faire collaborer efficacement et d'une manière cohérente les différents domaines d'expertise impliqués dans le projet de conception mécatronique ? (Axe 1)
— Q2 : Quelles sont les connaissances à échanger entre les différents modèles métiers ? (Axe 2)
— Q3 : Commet détecter et gérer les conflits entre les différents modèles métiers ? (Axe 3)
— Q4 :Comment choisir la solution qui satisfait au mieux les exigences ? (Axe

TABLE 1.5 – Caractérisation des entrées proposée par Guitouni et al. [97]

Input	**Signification**
I1	n structures { P_j, I_j } de préordres
I3	n structures { P_j, I_j} de semiordres et/ou structures { P_j, I_j, Q_j} de pseudo-ordres plus des seuils de véto v_j
I4	n fonctions d'utilité (u_j) (d'utilité partielle) exprimées sur des échelles intervalles
I10	n structures { P_j, I_j} de semiordres et/ou structures { P_j, I_j, Q_j} de pseudo-ordres et une relation de pré-ordres complet($>$) sur les attributs (critères)
I12	n fonctions d'utilité (u_j) (d'utilité partielle) exprimées sur des échelles intervalles et une relation de pré-ordres complet($>$) sur les attributs
I16	n structures { P_j, I_j} de semiordres et/ou structures { P_j, I_j, Q_j} de pseudo-ordres plus des seuils de véto v_j sur des évaluations distributionnelles plus en relation de pré-ordres complet($>$) sur les attributs
I17	n structures { P_j, I_j } de préordres plus vecteur de coefficients d'importance relative (II) des attributs
I18	n structures { P_j, I_j} de semiordres et/ou structures { P_j, I_j, Q_j} de pseudo-ordres plus un vecteur de coefficients d'importance relative (II) des attributs
I19	n structures { P_j, I_j} de semiordres et/ou structures { P_j, I_j, Q_j} de pseudo-ordres plus des seuils de véto v_j plus un vecteur de coefficients d'importance relative (II) des attributs
I20	n fonctions d'utilité (u_j) (d'utilité partielle) exprimées sur des échelles intervalles plus un vecteur de coefficients d'importance relative (II) des attributs
I21	n fonctions d'utilité (u_j) (d'utilité partielle) exprimées sur des échelles ratios plus un vecteur de coefficient d'importance relative (II) des attributs

4)

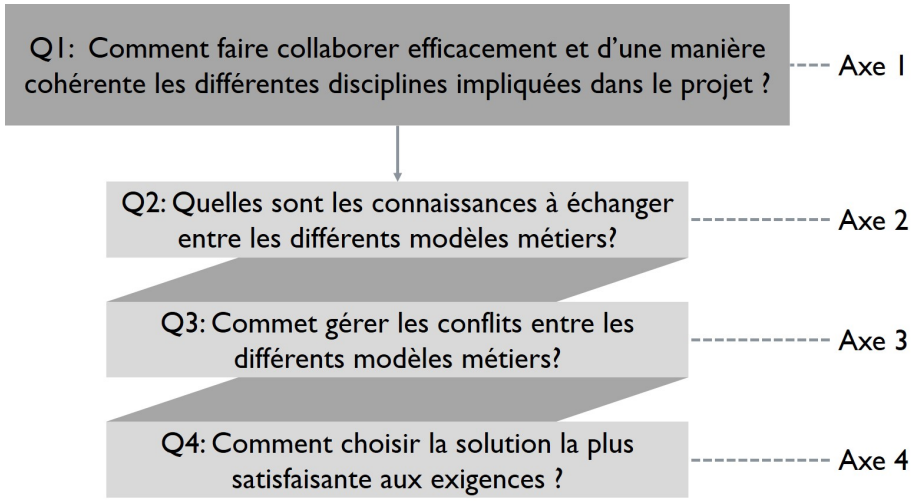


FIGURE 1.22 – Les questions de recherche et les axes correspondants

1.10 Vers un cadre mathématique formel

1.10.1 Besoin d'un outil mathématique formel

La conception des systèmes mécatroniques englobe plusieurs domaines d'expertise liés les uns aux autres, ce qui rend la tâche de conception assez complexe. Les domaines d'expertise doivent échanger leurs connaissances afin d'assurer une collaboration efficace. Cependant, compte tenu de la grande quantité de connaissances utilisées dans le processus de conception collaborative, seules les connaissances cruciales doivent être partagées et ensuite capitalisées. Par conséquent, l'utilisation des solutions informelles pour identifier les connaissances les plus importantes peut être chronophage et nécessite également un investissement humain important. Par conséquent, pour optimiser l'identification des connaissances cruciales, des méthodes formelles basées sur des techniques mathématiques deviennent nécessaires.

En outre, au cours de la collaboration, plusieurs conflits peuvent apparaître et doivent être soigneusement détectés et gérés. Dans les modèles de collaboration précédents, la gestion des conflits est basée sur les expériences et les connaissances antérieures du chef de projet. Dans ces modèles, un conflit est considéré comme une incompatibilité entre deux ou plusieurs valeurs d'un paramètre et lorsque ces valeurs ne respectent pas les contraintes définies. Les dépendances entre les paramètres sont également négligées, ce qui peut influencer la fiabilité des décisions. Par conséquent, un cadre formel pour gérer les conflits est nécessaire.

Dans ce contexte, la théorie des catégories (TC) peut être considérée comme un cadre efficace et formel pour traiter l'identification des connaissances cruciales

et la gestion des conflits. Cette théorie a récemment attiré l'attention en tant que formalisme potentiel pour la conception collaborative et l'ingénierie système [98]. Par conséquent, nous proposons dans ce travail de recherche un nouveau modèle de collaboration pour le partage des connaissances enrichi par les constructions puissantes de la TC. Dans la section suivante, la théorie des catégories ainsi que ses notions de base seront présentées.

1.10.2 Théorie des catégories

Le Théorie des catégories est une théorie mathématique introduite par Samuel Eilenberg et Saunders Mac Lane à la fin des années 1940 [98]. Cette théorie a été développée pour relier deux domaines distincts : l'algèbre et la topologie. Elle vise à décrire la relation entre les objets plutôt que les objets eux-mêmes et à ignorer leur structure interne. Avec cette théorie, il est possible de formaliser une idée ou un concept en utilisant une catégorie et en reliant cette catégorie à une autre par des foncteurs [99].

Une **catégorie** est composée d'un ensemble d'**objets** X, Y, Z, etc. et un ensemble de **flèches** ou **morphismes** f, g, h, etc. Chaque morphisme f a un **domaine** (dom) et un **co-domaine** (cod) f : X $\rightarrow$ Y, où X = dom(f) et Y = cod(f). Étant donné deux morphismes f : X $\rightarrow$ Y et g : Y $\rightarrow$ Z, il existe un morphisme donné g $\circ$ f : X $\rightarrow$ Z appelé la composition de g avec f (voir figure 1.23.a). De plus, pour chaque objet X, il existe un morphisme 1_X : X $\rightarrow$ X, appelé morphisme d'identité en respectant la propriété suivante : f $\circ$ 1_X = f = 1_Y $\circ$ f [100].

Un **foncteur** F : $\mathcal{C} \Longrightarrow \mathcal{D}$ entre deux catégories $\mathcal{C}$ et $\mathcal{D}$ est un mapping d'objets en objets et de morphismes en morphismes (voir figure 1.23.a).

Une catégorie **discrète** est une catégorie où tous les morphismes sont des morphismes d'identité. La composition et l'identité ne peuvent pas être satisfaites puisqu'il n'existe que des flèches d'identité [100].

Une catégorie **tranche** ou *Comma category* est une construction dans la TC établie à partir de deux foncteurs avec le même co-domaine. Ce concept permet de faire varier l'objet source et l'objet cible sur l'image de deux foncteurs distincts [100]. Soit $\mathcal{A}$, $\mathcal{B}$ et $\mathcal{C}$ des catégories et F : $\mathcal{A} \Longrightarrow \mathcal{C}$ et G : $\mathcal{B} \Longrightarrow \mathcal{C}$, soient deux foncteurs avec le même co-domaine. La catégorie tranche peut être formée comme suit : les objets sont tous des triples (X, Y, h) où X est un objet dans $\mathcal{A}$, Y un objet dans $\mathcal{B}$ et h : F($\mathcal{A}$) $\rightarrow$ G($\mathcal{B}$) un morphisme dans $\mathcal{C}$. Les morphismes de (X, Y, h) vers (X', Y' et h') sont tous des paires (α, β), où α : X $\rightarrow$ X' et β : Y $\rightarrow$ Y' sont des morphismes dans $\mathcal{A}$ et $\mathcal{B}$ respectivement, pour lesquels le carré commute, c'est-à-dire G(β) $\circ$ h = h' $\circ$ F(α) (voir figure 1.23.b).

Dans une catégorie $\mathcal{C}$, un **Pullback** de flèches f et g avec cod(f)= cod(g) est

composé de P1 et P2 telles que f ∘ P1= g ∘ P2. Étant donné tout morphisme w_1 : W → X et w_2 : W → Y avec f ∘ w_1= g ∘ w_2, il existe un unique u : w → P avec w_1 = P_1 ∘ u et w_2= P_2 ∘ u (voir figure 1.23.c). Le dual d'un pullback peut être défini comme un **Pushout**.

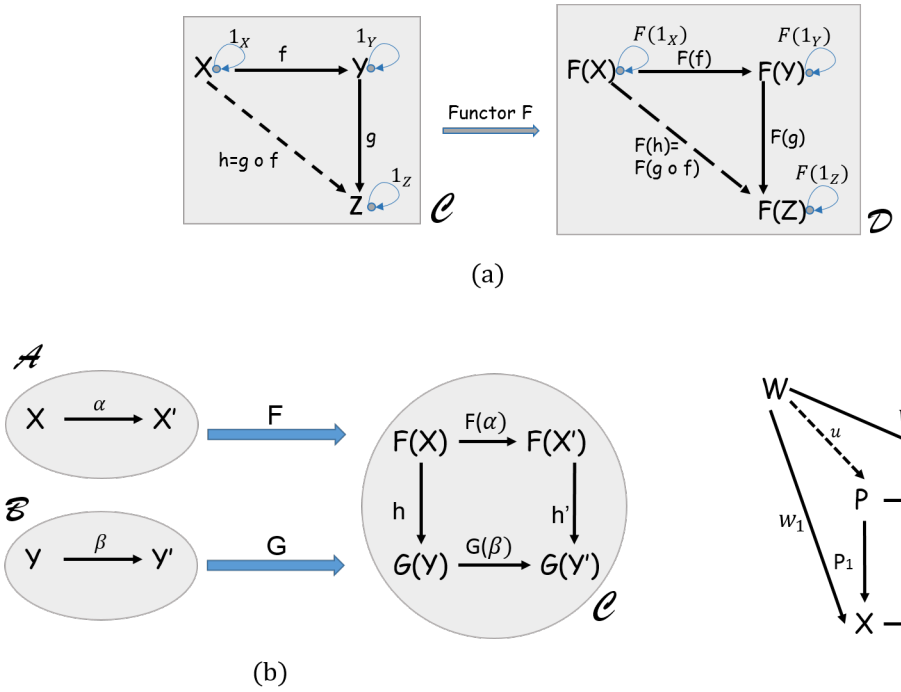


FIGURE 1.23 – Concepts fondamentaux de la théorie des catégories [100]

1.10.3 Méthodes basées sur la théories des catégories dans le contexte de la conception collaborative et l'ingénierie système

La théorie des catégories a gagné un intérêt croissant en tant que formalisme potentiel dans les domaines de la conception collaborative et de l'ingénierie système [101]. Différentes approches ont été proposées dans ce contexte.

Une approche basée sur la TC dans l'ingénierie système basée sur les modèles (MBSE) a été présentée par Mabork Ryan [98]. Dans cette approche, une base formelle pour le MBSE utilisant des concepts de la TC est proposée. L'idée est de considérer le système entier comme une catégorie et ses éléments comme des objets. Une autre approche intéressante a été proposée par Ormandjieva et al. [82], qui consiste à construire un modèle unifié de systèmes multi-agents. Au moyen de la théorie des catégories, la communication entre les agents peut être atteinte. Cette

approche vise à vérifier les propriétés du système par le biais des catégories et des foncteurs. Par ailleurs, un cadre catégorique pour concevoir et mettre en œuvre des systèmes concurrents a été proposé dans [102, 103]. Les foncteurs sont utilisés dans ce cadre comme un outil de vérification de la cohérence entre la conception et l'implémentation. De plus, Kibret et al. [104] ont présenté une modélisation des systèmes par catégories. Les systèmes sont considérés comme des catégories et leurs parties constituantes sont représentées comme des objets. En outre, un cadre intéressant a été introduit par [101]. Les auteurs ont présenté un cadre global basé sur la TC pour la transformation des modèles. Le cadre *Concept-Model-Graph-View Cycle* (CMGVC) est introduit afin de faciliter une analyse cohérente de l'architecture.

Comme nous pouvons le remarquer dans les approches présentées précédemment, la TC est utilisée comme un formalisme commun pour décrire les systèmes complexes. De même, cette théorie sera utilisée dans notre travail de recherche pour représenter les données de granularité fine utilisées dans la conception collaborative mécatronique. Cette théorie permettra l'identification des connaissances cruciales ainsi que la détection et la gestion des conflits.

1.11 Conclusion

Dans ce premier chapitre, on est parti de quelques généralités sur les systèmes mécatroniques ainsi que les différents processus de conception. Par la suite, après avoir passé en revue les différentes solutions utilisées pour supporter la conception collaborative des systèmes mécatroniques, nous avons constaté qu'une méthode formelle pour extraire les connaissances cruciales est nécessaire. Pour cela, une étude détaillée sur les méthodes d'identification des connaissances cruciales est présentée dans ce chapitre. Par ailleurs, nous avons remarqué que les supports de collaboration ne présentent pas des solutions pour détecter et gérer les conflits. De ce fait, une revue de littérature a été présentée dans ce contexte. Un résumé sur les différentes méthodes d'aide à la décision multicritère utilisées dans la conception mécatronique a été également précisé.

L'étude bibliographique présentée dans ce chapitre nous a conduit à définir notre problématique ainsi que les questions de recherche. Étant donnée sa base formelle mathématique utile pour extraire les connaissances cruciales et gérer les conflits, la théorie des catégories ainsi que ses concepts fondamentaux ont été détaillés. Cette théorie sera considérée comme un outil formel pour notre méthodologie générale que nous allons présenter dans le chapitre suivant.

Chapitre 2

Méthodologie générale pour la collaboration et l'aide à la décision

Après avoir passé en revue les différentes méthodologies relatives à la collaboration et au partage des connaissances, nous proposons dans ce chapitre une méthodologie générale de collaboration et d'aide à la décision basée sur la théorie des catégories que nous appelons *Category Theory-based Collaborative Design and Decision-making* (CaTCoDD) *methodology*. On commence par présenter les différentes étapes de notre méthodologie CaTCoDD. Ensuite, on détaille les concepts fondamentaux de cette méthodologie. A la fin, on décrit le méta-modèle et le démonstrateur associé pour illustrer notre proposition.

2.1 Terminologie : donnée, information et connaissance

La connaissance est définie selon Sanchez & Heene [105] comme une idée plus ou moins complète de quelque chose. Une confusion entre la connaissance, l'information et la donnée est toujours présente. Pour cela, il est nécessaire de préciser la différence entre ces trois notions même si elles sont interdépendantes.

Une donnée est un fait objectif où aucun sens particulier ou contexte ne sont effectués. Dans le contexte de la conception collaborative, Badin [44] a proposé une définition de la donnée comme étant un paramètre utilisé dans un modèle métier qui ne reflète aucun sens particulier. Par exemple, une chaîne de caractère "Diamètre" ou une valeur "30" extraites d'un modèle métier sont considérées comme des données.

Par ailleurs, une information peut être construite après avoir accordé un sens bien précis à une ou plusieurs données. Selon Badin [44], également, l'information est définie comme un résultat d'une donnée organisée et structurée et qui possède un contexte particulier d'utilisation. En accordant un contexte à l'exemple précédent, l'expression "Diamètre= 30mm" est considérée comme une information.

Enfin, une interprétation personnelle d'une information dans un contexte particulier est suffisante pour obtenir une connaissance. Par conséquent, la connaissance est issue d'une information, elle même obtenue à partir d'une donnée. Badin [44] affirme que l'ensemble des paramètres et contraintes utilisés dans un modèle métier durant la conception collaborative constitue une connaissance. Considérons l'expression suivante comme un exemple, "Le boîtier papillon doit avoir un diamètre = 30mm". Cette expression devient une connaissance pour un concepteur des éléments mécatroniques.

2.2 Typologie des connaissances

Nonaka & Takeuchi [106] et Grundstein [107] ont présenté une distinction fondamentale entre deux types de connaissances :

— Connaissances implicites/tacites : se caractérisent par leur aspect personnel puisqu'elles sont créées à partir du savoir-faire et de l'expérience de l'être humain. Ce genre de connaissances est difficile à articuler ou communiquer aux autres individus vu leur aspect personnel.
— Connaissances explicites : ces connaissances peuvent être clairement transmises et formalisées en ayant recours à des langages formels et structurés, par opposition aux connaissances implicites. Considérant leur aspect tangible, il est possible de réutiliser ses connaissances par d'autres individus.

Selon Nonaka & Takeuchi, la connaissance est une entité dynamique construite à partir de différentes conversions entre les connaissances implicites et explicites. Quatre modes de conversion sont envisagés comme représenté par la figure (2.1) :

- La socialisation : consiste à partager des expériences entre des individus pour créer ainsi des nouvelles connaissances tacites à travers des modèles mentaux partagés.
- L'externalisation : dans ce mode de conversion, les connaissances implicites sont formalisées pour avoir une forme explicite et compréhensible pour les autres personnes.
- L'intériorisation : c'est un processus d'apprentissage pour passer d'une connaissance explicite vers une connaissance implicite.
- La combinaison : elle s'agit de créer des connaissances explicites à travers une restructuration des connaissances explicites.

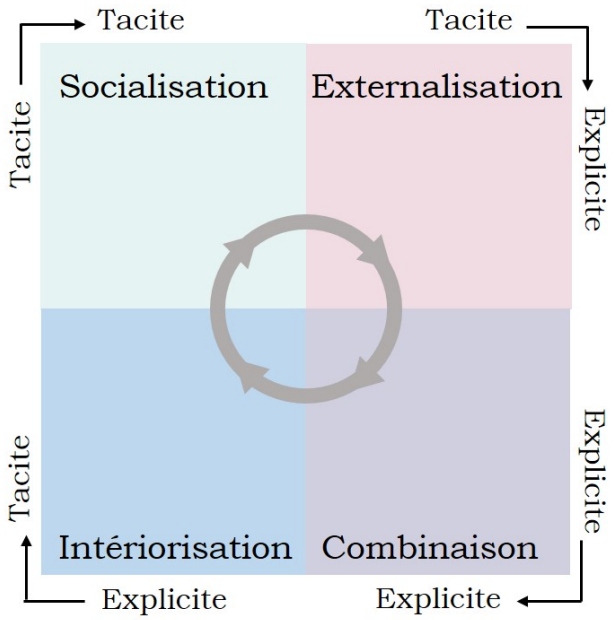


FIGURE 2.1 – Les interactions entre les connaissances explicites et implicites [106]

2.3 Processus de gestion des connaissances (*Knowledge management* KM)

La gestion des connaissances fait référence à l'identification et à l'exploitation des connaissances collectives au sein d'une organisation pour l'aider à être compétitive [108]. Ce concept implique des processus de création, stockage, récupération, transfert et application des connaissances [109]. La création des connaissances reflète la

créativité d'une organisation pour développer de nouveaux produits, nouvelles idées et nouvelles compétences, alors que le stockage et la récupération des connaissances consiste à conserver et structurer les connaissances d'une organisation dans des documentations écrites et dans des bases de données. Comme son nom l'indique, le transfert des connaissances revient à distribuer les connaissances d'une organisation aux endroits où elles peuvent être utilisées. Enfin, l'application des connaissances consiste à employer les connaissances afin de gérer des nouvelles situations au sein d'une organisation. Ce concept représente un facteur majeur dans les organisations pour faire face à une concurrence toujours croissante. [109].

La gestion des connaissances englobe deux concepts fondamentaux distincts mais interdépendants qui sont l'ingénierie des connaissances et l'ingénierie basée sur les connaissances comme nous pouvons le constater sur la figure (2.2).

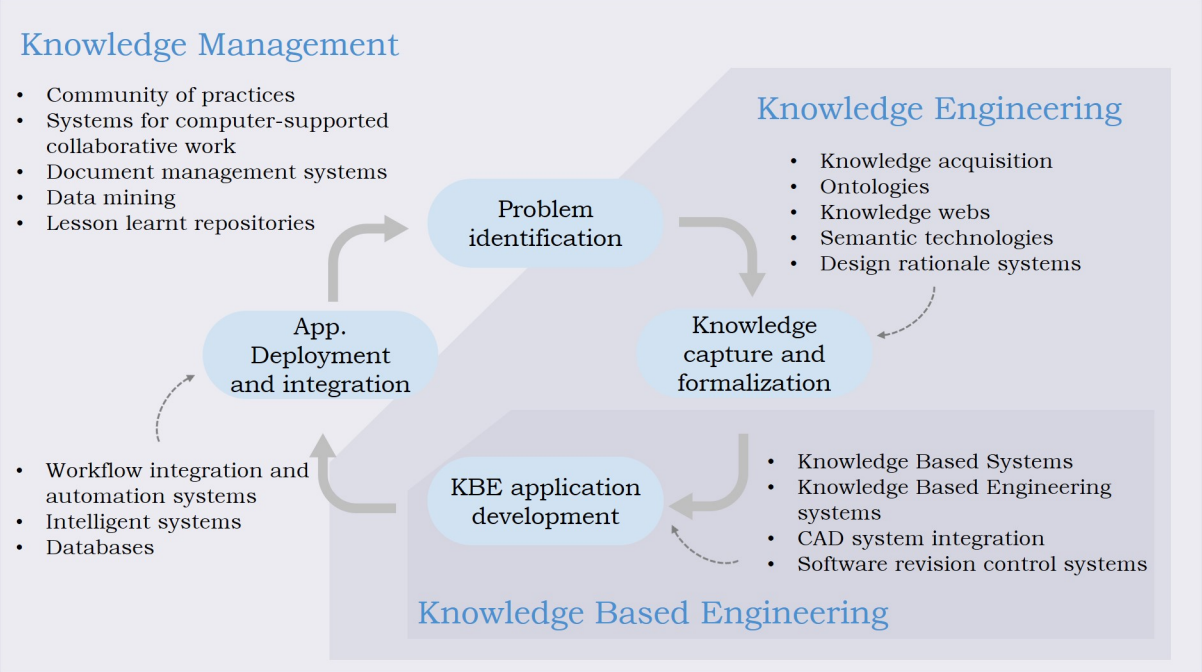


FIGURE 2.2 – Positionnement de la gestion des connaissances (KM), l'ingénierie des connaissances (KE) et l'ingénierie basée sur les connaissances (KBE) [110]

— Ingénierie des connaissances *(Knowledge Engineering KE)* : ce concept a recours à des méthodes particulières pour recueillir et structurer les connaissances [111]. Les ontologies sont utilisées dans l'ingénierie des connaissances dans la phase d'identification du problème.
— Ingénierie basée sur les connaissances *(Knowledge-Based Engineering KBE)* : peut être définie comme étant une technologie basée sur l'utilisation d'outils logiciels dans le but de capturer ainsi que réutiliser systématiquement la

connaissance. Cette technologie vise à réduire le coût et le temps de développement des produits à travers l'automatisation des tâches de conception et la prise en considération de l'optimisation multidisciplinaire [110].

2.4 La méthodologie CaTCoDD :Category Theory-based Collaborative Design and Decision-making

Comme on l'a souligné au chapitre précédent, le modèle *Collaborative Design Process and Product Knowledge* (CDPPK) a démontré son efficacité en termes de collaboration, de capitalisation et réutilisation des connaissances. Ce modèle est basé sur les concepts suivants (voir figure 2.3) :

— Le premier concept est le « *Design Process Knowledge* » (DPK) qui contient toutes les ICEs utilisées dans le projet. A ce niveau, les ingénieurs peuvent ajouter les paramètres et les contraintes nécessaires. Cette entité est considérée comme un espace de stockage d'information. Lorsque la collaboration commence, les ingénieurs participant à réalisation du projet peuvent se référer à cette base. A la fin de la collaboration, les ICEs contiendront les résultats finaux du projet.

— Le deuxième concept est nommé « *User Process Configuration*» (UPC). Cette configuration contient les instances d'ICEs. Dans le modèle CDPPK, les ICEs sont décomposées ce qui donne la flexibilité aux ingénieurs disciplinaires dans le choix des paramètres dont ils ont besoin.

— Un autre concept du modèle CDPPK est le «*Collaborative Design Process*» (CDP). Ce concept englobe toutes les UPCs du projet et permet de gérer les conflits entre les ingénieurs disciplinaires. Toute l'équipe du projet est en relation avec la CDP ce qui assure la traçabilité de la collaboration. Le processus de collaboration est représenté sous forme de graphes afin de faciliter la réutilisation des connaissances.

— Le dernier concept dans le modèle CDPPK, est le domaine du projet ou « *Project Domain*» (PD). Ce domaine représente l'environnement global dans lequel le système peut être défini et décomposé en plusieurs sous-systèmes.

Cependant, le modèle CDPPK présente certaines limites :

— Les connaissances qui doivent être capitalisées, appelées connaissances cruciales, doivent être identifiées afin d'assurer un échange et un partage efficaces entre les différents ingénieurs impliqués dans la collaboration. Afin d'identifier ces connaissances, un échange étroit entre les acteurs du projet est suggéré dans le modèle CDPPK. Cette solution est informelle, ce qui peut être considérablement chronophage et peut être difficile à organiser dans un environnement distribué comme celui de la conception des systèmes mécatro-

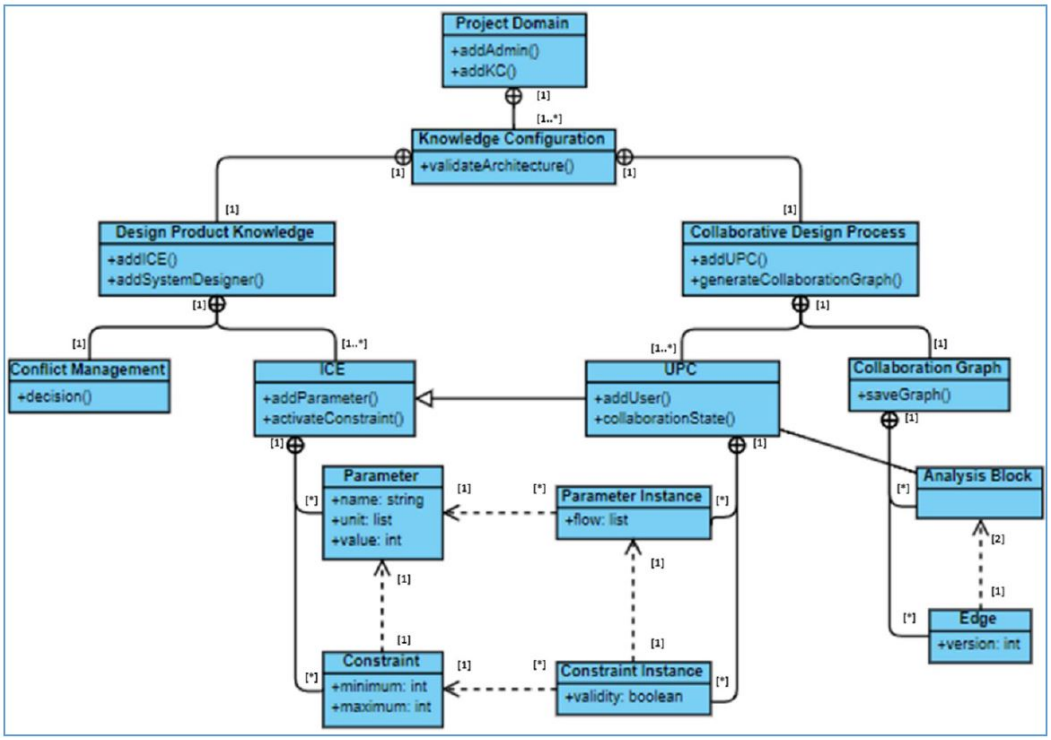


FIGURE 2.3 – Le méta-modèle de CDPPK [7]

niques.

— Les conflits survenant au cours de la collaboration dans le modèle CDPPK sont résolus à l'aide d'un processus manuel. Ce modèle ne présente pas des solutions explicites pour localiser et gérer ces conflits. De plus, la dépendance entre les paramètres est négligée, ce qui est critique pendant le processus de résolution des conflits. Chaque décision pour gérer un conflit doit prendre en compte le lien de dépendance entre les paramètres et la propagation des changements.

— Le modèle CDPPK traite une seule architecture d'un système mécatronique sans tenir en compte qu'un système peut présenter différentes architectures ou alternatives nécessitant la comparaison afin de choisir l'architecture qui satisfait au mieux les exigences.

Ainsi, les équipes de conception ont des difficultés à identifier les connaissances cruciales, à gérer les conflits et à comparer différentes architectures du système à concevoir. Notre objectif est de proposer alors un nouveau modèle de collaboration qui conserve les caractéristiques utiles du modèle CDPPK en termes de capitalisation des connaissances tout en surmontant les limites mentionnées ci-dessus. En ce sens, on propose une nouvelle méthodologie pour la conception collaborative et l'aide à la décision pour les systèmes mécatroniques nommée CaTCoDD.

2.5 Généralités

En se basant sur les quatre questions de recherche proposées dans le cadre du premier chapitre, on définit quatre axes de recherche qui permettent d'apporter des réponses à ces questions et qui traitent des problématiques particulières. Une description étendue de ces axes est fournie dans ce qui suit.

2.5.1 Axe 1 : Support de collaboration et partage des connaissances

Comme on l'a souligné précédemment, un support de collaboration prenant en compte à la fois, l'identification formelle des connaissances cruciales, la gestion des conflits et le choix d'architectures est nécessaire afin de partager d'une manière efficace les connaissances entre les différentes parties prenantes d'un projet de conception mécatronique. De ce fait, ce premier axe met en valeur notre méthodologie générale CaTCoDD. Cet axe regroupe trois sous-axes, notamment les axes 2, 3 et 4 (voir figure 2.4).

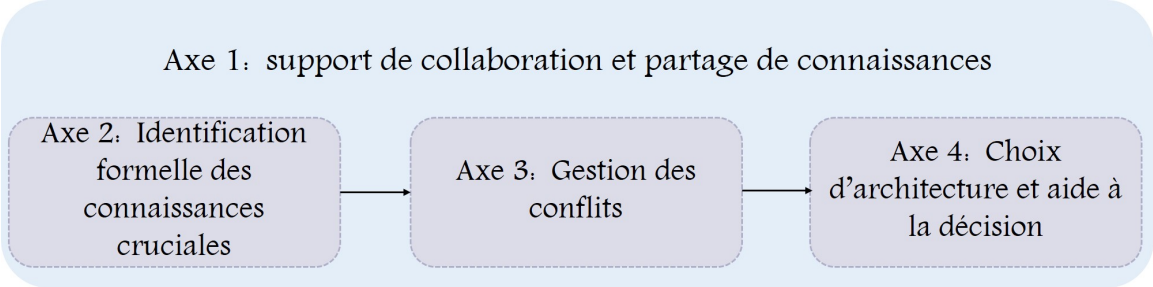


FIGURE 2.4 – Positionnement de l'axe 2, 3 et 4 dans le premier axe de recherche

2.5.2 Axe 2 : Identification des connaissances cruciales

Le deuxième axe concerne l'identification des connaissances cruciales. Comme nous pouvons le remarquer à partir de l'étude bibliographique précédente, l'extraction de ces connaissances s'effectue d'une façon informelle en se basant sur des échanges et des discussions entre les participants à la conception. Cependant, cela peut être chronophage et peut être difficile dans le cas d'un environnement complexe et distribué. De ce fait, la formalisation de cette étape devient nécessaire. Dans ce contexte, on proposera d'utiliser la théorie des catégories comme étant une base formelle pour l'identification des connaissances. En se basant sur la représentation catégorique de l'ensemble des connaissances, on peut extraire d'une manière formelle les connaissances utiles à la collaboration.

2.5.3 Axe 3 : Gestion des conflits

Dans le troisième axe de recherche, nous allons apporter une réponse à la question : "comment détecter et gérer les conflits entre les différents modèles métiers ?" Dans cet axe, une méthode basée sur la théorie des catégories sera proposée afin de détecter ainsi que résoudre les conflits apparus durant la conception collaborative. Cette méthode est basée également sur les règles de cohérences et les *patterns* pour mettre en valeur les conflits qui peuvent avoir lieu et pour faciliter ainsi la détection de ces incohérences. Ces règles de cohérences rendent le processus de gestion des conflits flexible en ajoutant, modifiant ou même supprimant des règles à chaque niveau du processus.

2.5.4 Axe 4 : Aide à la décision pour le choix d'architectures

Le dernier axe de notre travail de recherche concerne la comparaison entre différentes architectures d'un système mécatronique. Étant donnée la nature multidisciplinaire des systèmes mécatroniques, différentes solutions ou architectures peuvent se présenter pour le même système. De ce fait, une comparaison entre ces alternatives est primordiale. Afin d'effecteur cette comparaison d'une manière objective, le recours aux méthodes d'aide à la décision multicritère fera l'objet du dernier axe.

Les axes de recherche identifiés ci-dessus se présentent dans notre méthodologie comme nous allons le constater dans ce qui suit.

2.6 Étapes de la méthodologie CaTCoDD

La conception mécatronique fait intervenir plusieurs disciplines. La diversité de ces disciplines peut entraîner des incohérences entre les parties prenantes. Par conséquent, la collaboration entre les domaines hétérogènes devient nécessaire. Nous proposons ainsi une nouvelle méthodologie adaptée à l'identification des connaissances cruciales, la résolution des conflits et le choix d'architectures. Notre nouvelle approche est illustrée par la figure (2.5). Cette approche peut être résumée en trois étapes principales. La première étape est l'initialisation du projet de conception, autrement dit c'est une étape de pré-collaboration. La deuxième étape concerne la collaboration où l'échange et le partage entre les différents ingénieurs auront lieu. La dernière étape est consacrée au choix d'architectures. Les acteurs qui interviendront dans cette méthodologie sont : le chef de projet, l'ingénieur système et les différents ingénieurs disciplinaires.

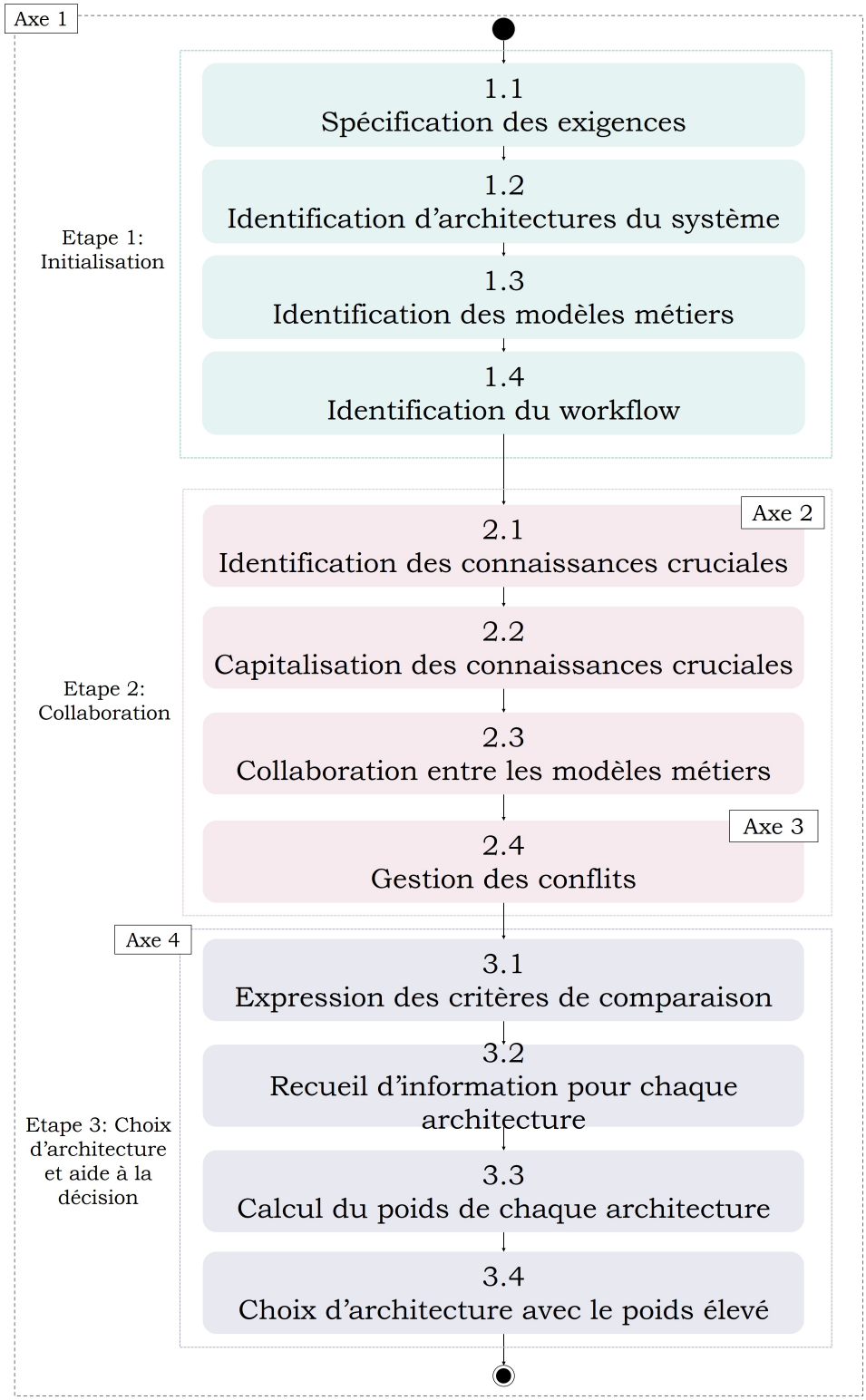


FIGURE 2.5 – Les différentes étapes de la méthodologie CaTCoDD

2.6.1 Étape 1 : Initialisation

Cette première étape est une étape de pré-collaboration ou initialisation dans laquelle le chef de projet ainsi que l'ingénieur système préparent les éléments dont les ingénieurs disciplinaires ont besoin afin d'assurer la collaboration et l'échange des connaissances. Le chef de projet commence cette étape par analyser les différentes exigences du client qui doivent être respectées par tous les participants au projet de conception. Le rôle du chef de projet consiste essentiellement à mener toute l'équipe vers la réalisation satisfaisante du projet de conception. Une fois les exigences ont été bien définies, l'ingénieur système propose les différentes architectures ou alternatives du système à concevoir.

La différence entre ces alternatives peut concerner la géométrie, le type de motorisation, le type d'énergie, le degré d'intégration des composants, etc. En se basant sur les architectures choisies, les modèles métiers (ou *Expert Models*) pour chaque architecture doivent être identifiés par l'ingénieur système. Ces modèles sont établis par différents acteurs et concernent des domaines d'expertise hétérogènes. Ils peuvent se présenter sous différentes formes : un modèle dynamique (modèle multi-physique établi à l'aide d'un outil de modélisation et de simulation multi-physique), un modèle 3D (modèle géométrique crée par un logiciel de CAO (Conception Assistée par Ordinateur)), un modèle fluide (basé sur les méthodes des éléments finis), etc. Même si notre objectif consiste à supporter la collaboration entre les différents modèles métiers, il est important de prendre en considération un ordre bien défini lors de la création de chaque modèle métier. Pour cela, le *workflow* de la conception doit être identifié à la fin de l'étape d'initialisation de notre méthodologie. L'achèvement de cette première étape enclenche la deuxième étape, où les connaissances cruciales seront extraites et échangées entre les ingénieurs disciplinaires.

2.6.2 Étape 2 : Collaboration

Compte tenu que cette étape intermédiaire s'articule, essentiellement, autour la collaboration, les différents ingénieurs disciplinaires doivent intervenir à ce niveau. La première sous-étape consiste à extraire les connaissances cruciales nécessaires à la collaboration. Cette tâche représente la clé de la réussite de la conception collaborative. Autrement dit, seulement les connaissances qui ont un rôle dans la collaboration doivent être partagées et capitalisées. Les approches citées dans le chapitre précédent ont montré leur limite en termes d'identification formelle des connaissances cruciales. De ce fait, on propose dans cette tâche d'avoir recours à la théorie des catégories comme étant un outil de formalisation des connaissances cruciales. Ceci va permettre d'avoir une vue globale des connaissances nécessaires à la collaboration ainsi qu'une représentation formelle compréhensible pour tous les ingénieurs disciplinaires. En se basant sur la représentation catégorique, l'ingénieur

système capitalise les connaissances cruciales sous forme d'entités dans le support de collaboration. Ensuite, les ingénieurs disciplinaires commencent le développement de leurs modèles métiers tout en collaborant ensemble et échangeant leurs connaissances. Cet échange implique l'apparition de conflits potentiels entre les ingénieurs disciplinaires. Ces conflits doivent être détectés et gérés pour assurer la cohérence du système à concevoir. La gestion des conflits est basée sur la théorie des catégories comme on l'a déjà souligné précédemment. A ce niveau, la deuxième étape de notre méthodologie est accomplie et on peut passer à la dernière étape où un processus d'aide à la décision sera utilisé afin d'aider le chef de projet à choisir l'architecture qui satisfait au mieux les exigences spécifiées dès la première phase de notre méthodologie.

2.6.3 Étape 3 : Prise de décision et choix d'architectures

Une fois les ingénieurs disciplinaires ont pu collaborer ensemble pour concevoir les différentes architectures du système mécatronique, l'aide à la décision devient nécessaire. En effet, en se basant sur les résultats de collaboration, l'ingénieur système peut effectuer une comparaison entre les différentes architectures afin de converger vers l'architecture la plus adéquate. Cette étape commence par l'expression de préférences, autrement dit, les critères nécessaires à la comparaison. Ensuite, un recueil d'information pour chaque architecture est effectué. Ces deux sous-étapes sont suivies ensuite du calcul de poids de chaque alternative en se basant sur la méthode d'aide la décision *Analytical Hierarchical Process* (AHP). L'architecture qui possède le poids le plus important représente la solution qui satisfait au mieux les exigences.

Grâce à la méthodologie CaTCoDD, les connaissances cruciales nécessaires à la collaboration peuvent être extraites de manière formelle en utilisant la théorie des catégories. Cette formalisation facilitera la résolution des conflits. De plus, notre proposition permet l'évaluation de différentes architectures d'un même système, contrairement aux méthodologies présentées dans la littérature. Nous présentons dans ce qui suit la structure et les concepts fondamentaux de CaTCoDD.

2.7 Structure et concepts fondamentaux de la méthodologie CaTCoDD

2.7.1 Structure de la méthodologie CaTCoDD

La méthodologie CaTCoDD est structurée comme le montre la figure (2.6). Quatre éléments principaux constituent notre méthodologie : le processus d'identification des connaissances cruciales (*Crucial Knowledge Identification Process)*, les

connaissances du produit (*Product Knowledge)*, le processus de gestion des conflits (*Conflict Resolution Process*) et le processus de choix d'architectures.

L'idée de la méthodologie CaTCoDD est de commencer par la formalisation de différents modèles métiers impliqués dans la collaboration. Cette unification se base sur la TC, où chaque paramètre est représenté sous forme d'une catégorie qui contient à son tour différentes instances provenant de différents modèles métiers. En se basant sur le formalisme de la TC, l'extraction des connaissances cruciales est possible. A ce niveau, le paramètre dont la catégorie est constituée de plusieurs instances, est considéré crucial. Uniquement les paramètres cruciaux seront partagés et capitalisés sous forme d'entités d'information. Une fois tous les paramètres capitalisés, la collaboration et le partage entre les ingénieurs disciplinaires peuvent commencer. Ce partage provoque un certain nombre de conflits qui doivent être détectés et gérés. Dans notre méthodologie, un conflit est défini comme une contradiction entre deux valeurs différentes du même paramètre ou lorsqu'un paramètre ne respecte pas une contrainte.

Pour commencer le processus de gestion des conflits, le formalisme basé sur la théorie des catégories, défini durant les premières phases de la méthodologie CaTCoDD, nécessite une mise à jour. Cette mise à jour réside dans l'enrichissement des catégories par les valeurs des paramètres obtenues après la collaboration. Cette étape est suivie de l'identification de dépendances entre tous les paramètres utilisés dans la collaboration.

La dépendance est définie sous forme d'une catégorie où les objets représentent les paramètres cruciaux et les morphismes illustrent le lien de dépendance entre ces paramètres. Tout changement dans un paramètre implique une modification dans les paramètres dépendants, pour cela, la dépendance entre les paramètres sera tenue en compte lors de la résolution des conflits. Pour mettre en valeur les contraintes qui doivent être respectées, un ensemble de règles de cohérences est établi. En se basant sur ces règles, des *patterns* sont définis sous forme de catégorie pour illustrer les conflits qui peuvent apparaître entre deux valeurs différentes d'un paramètre. Afin de localiser ces conflits, il suffit d'effectuer un mapping entre les catégories des patterns et les catégories des paramètres.

Une fois les conflits détectés, un ensemble d'actions de résolution doit être effectué afin de gérer ces conflits. Après avoir effectué des actions de résolution, une mise à jour des catégories des paramètres est nécessaire. Cette mise à jour sera suivie ensuite de la localisation des conflits de nouveau pour vérifier si la résolution d'un conflit n'a pas provoqué l'apparition d'un autre conflit. Ces étapes se répètent jusqu'à la résolution de tous les conflits apparus. Les valeurs finales obtenues à la fin du processus de gestion des conflits seront sauvegardées et utilisées dans le processus de choix d'architectures et aide à la décision.

Pour commencer la comparaison entre les différentes architectures, les critères

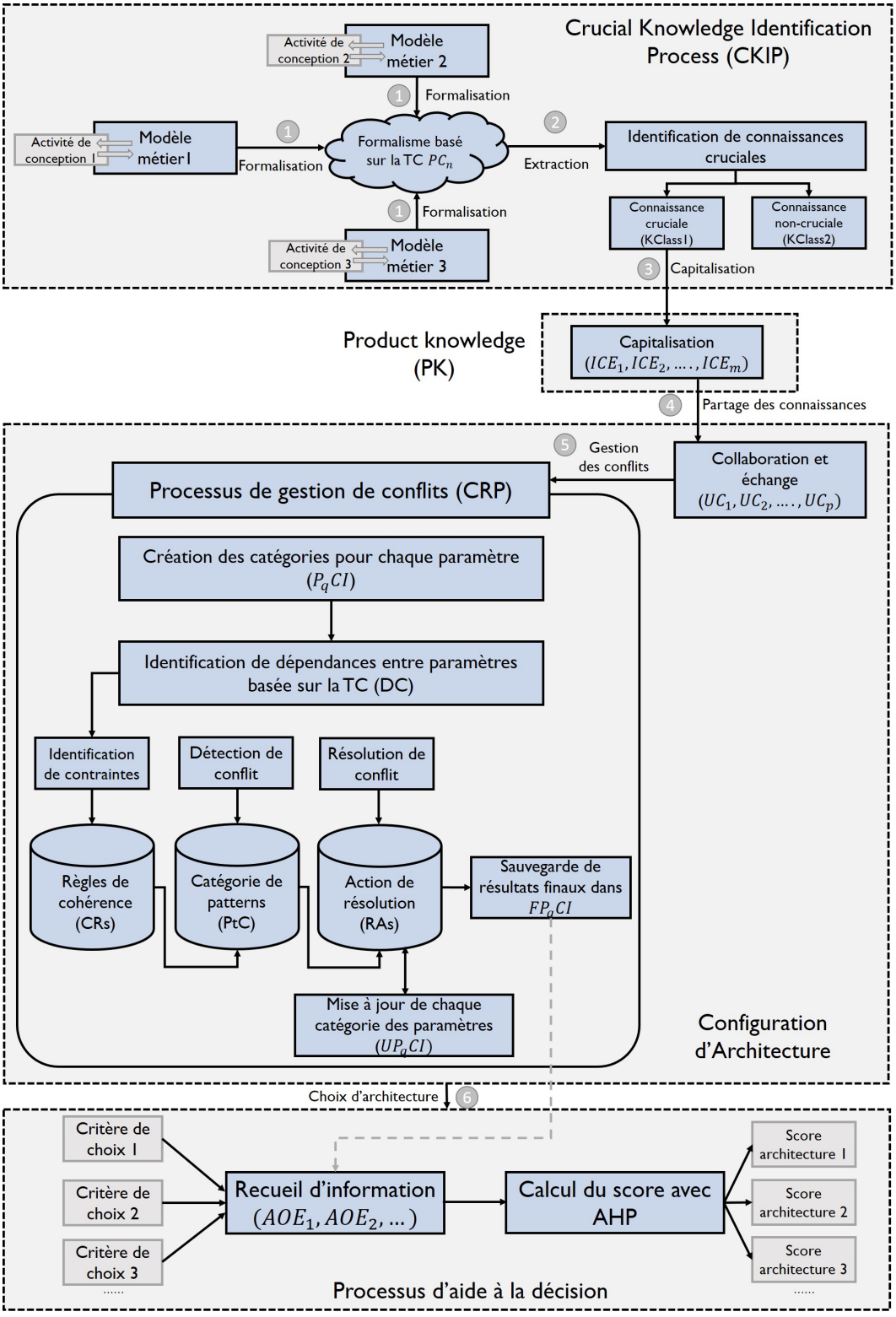


FIGURE 2.6 – Structure de la méthodologie CaTCoDD

de choix sont définis et les informations correspondantes pour chaque alternative sont recueillies à partir des résultats obtenus après la collaboration et la gestion des conflits. En ce qui suit, le score de chaque alternative doit être calculé pour aboutir finalement à un classement d'alternatives. Ce calcul s'effectue en se basant sur la méthode d'aide à la décision AHP. L'architecture avec le score le plus élevée sera considérée comme la solution qui satisfait au mieux les critères définis précédemment.

La méthodologie CaTCoDD permet de supporter à la fois, l'identification formelle des connaissances utiles à la collaboration, la localisation et la résolution des conflits ainsi que le choix d'architectures. Tous les concepts fondamentaux de cette approche seront plus détaillés ci-après.

2.7.2 Détail du concept *Expert Model (EM)*

Les modèles métiers ou *Expert models (EM)* font référence à un ensemble de modèles relatifs à différentes activités de conception et qui sont impliqués dans le processus de collaboration. Ces modèles peuvent prendre différentes formes telles que : un modèle multi-physique utilisant le langage Modelica dans l'environnement Dymola [5], un modèle de contrôle avec le logiciel Simulink, un modèle COTS (*Commercial Off-The Shelf*) pour sélectionner les composants appropriés avec les résultats obtenus par la simulation, etc.

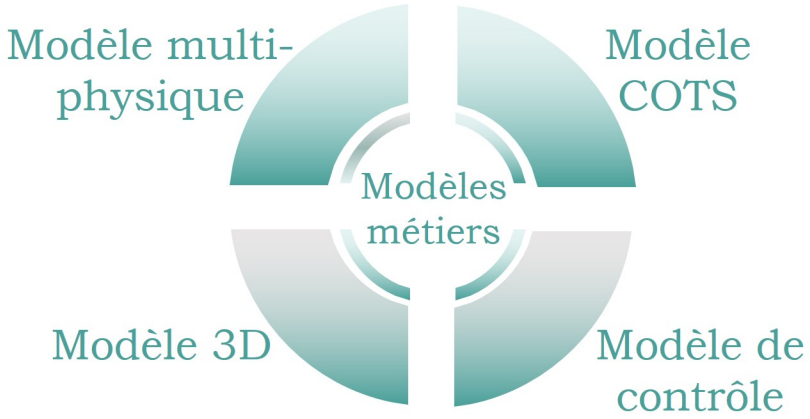


FIGURE 2.7 – Les modèles métiers dans le contexte de la conception collaborative

5. Dymola, Dassault Systèmes, `https://www.3ds.com/fr/produits-et-services/catia/produits/dymola/`

2.7.3 Détail du concept *Project Domain (PD)*

Le domaine de projet ou *Project Domain* représente l'environnement global de la conception où le système mécatronique est décomposé en sous-systèmes. Ce domaine regroupe toutes les activités de conception ainsi que les différentes architectures du système à concevoir.

2.7.4 Détail du concept *Crucial Knowledge Identification Process (CKIP)*

Le principe du processus d'identification des connaissances cruciales se base sur la théorie des catégories. C'est au niveau de ce processus que l'unification des modèles métiers hétérogènes a lieu. Comme on l'a expliqué précédemment, chaque paramètre crucial est considéré comme une catégorie. Chaque catégorie contient un certain nombre d'objets pour illustrer les différentes instances du paramètre correspondant. Les liens entre ces objets soulignent les activités de conception effectuées pour passer d'une instance vers une autre. Ce processus sera plus détaillé dans le chapitre suivant.

2.7.5 Détail du concept *Information Core Entity (ICE)*

Cette classe a été proposée précédemment dans [10, 7] comme une solution pour capitaliser les connaissances durant la collaboration. Ce concept sera également utilisé dans notre méthodologie. L'ICE peut être définie comme une entité décomposable qui regroupe les paramètres et contraintes nécessaires à la collaboration. Cette entité donne aux ingénieurs disciplinaires la flexibilité d'instancier seulement les paramètres dont ils ont besoin. L'ICE permet d'obtenir une base dynamique de connaissances tout au long du processus de conception. Chaque entité dispose d'un cycle de vie, de différentes versions et d'un niveau de criticité particulier.

L'ICE est composée essentiellement d'un ensemble de paramètres et contraintes. Le paramètre est défini, selon Badin [44], comme une donnée qui reflète un point de vue d'utilisation particulier. Ce paramètre est capitalisé soit sans lui attribuer une valeur d'une façon générique ou en lui attribuant une valeur par défaut qui peut être modifiée. Par ailleurs, une contrainte ou une règle métier est considérée comme une restriction basée sur des lois propres à une discipline. Cinq type de contraintes peuvent se présenter :

— Valeur nominale : dans ce cas, le paramètre doit porter uniquement la valeur nominale définie (e.g. Module d'Young).
— Bornes : A ce niveau, le paramètre peut porter des valeurs appartenant un domaine précis.

— Relations mathématiques : ce concept se base sur des opérateurs mathématiques pour exprimer la contrainte.
— Règles logiques : la relation entre paramètres est exprimée sous forme d'un type logique (e.g., if.... then...).
— Tableau de valeur discrète : dans ce cas, une valeur en entrée (dans une première colonne) correspond à une valeur propre dans un seconde colonne.

Dans notre approche CaTCoDD, seules les contraintes avec des bornes seront considérées dans la résolution des conflits.

Comme les ICEs ne sont pas des entités statiques, ces dernières possèdent un cycle de vie propre à eux indépendamment des modèles métiers. La figure (2.8) illustre le cycle de vie d'une ICE. Ce cycle est constitué de trois étapes principales. Dans la première étape (*Work in Progress*) l'ICE est créée dans sa version zéro V0, à ce niveau l'ICE peut être modifiée mais ne peut pas être utilisée ou instanciée dans les configurations d'utilisateurs. L'étape suivante est la publication. Dès que l'ICE est publiée, on peut l'instancier et même la modifier (diverses versions peuvent être créées et qui seront toutes enregistrées). Une fois l'ICE est archivée, elle devient obsolète où aucune instanciation n'est plus possible.

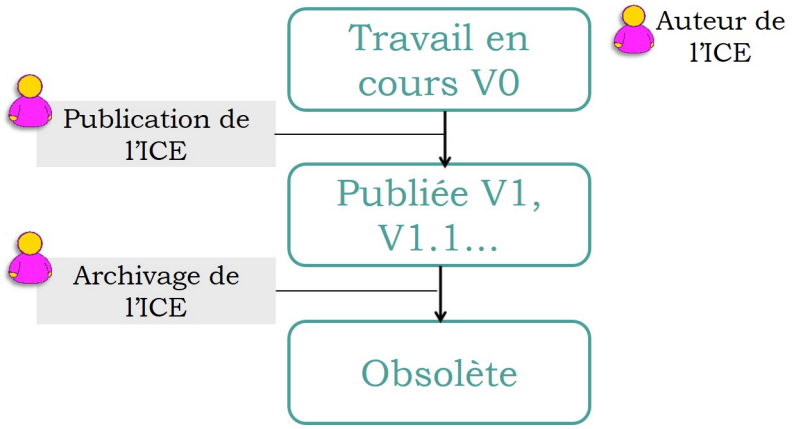


FIGURE 2.8 – Cycle de vie d'une ICE

2.7.6 Détail du concept *Knowledge Configuration (KC)*

Ce concept a été introduit par Badin et al. [10] et représente une entité qui regroupe un ensemble de connaissances structurées pour un objectif donné en relation avec un ou plusieurs modèles métiers. KC a été également utilisée par Mcharek et al. [7] pour regrouper les ICEs. Ce concept sera appliqué dans notre travail de recherche pour le même objectif. Cependant, elle est enrichie par les nouvelles classes pour l'identification des connaissances cruciales, la gestion des conflits et le choix

d'architectures. La structure de KC dans la méthodologie CaTCoDD est illustrée par la figure (2.9).

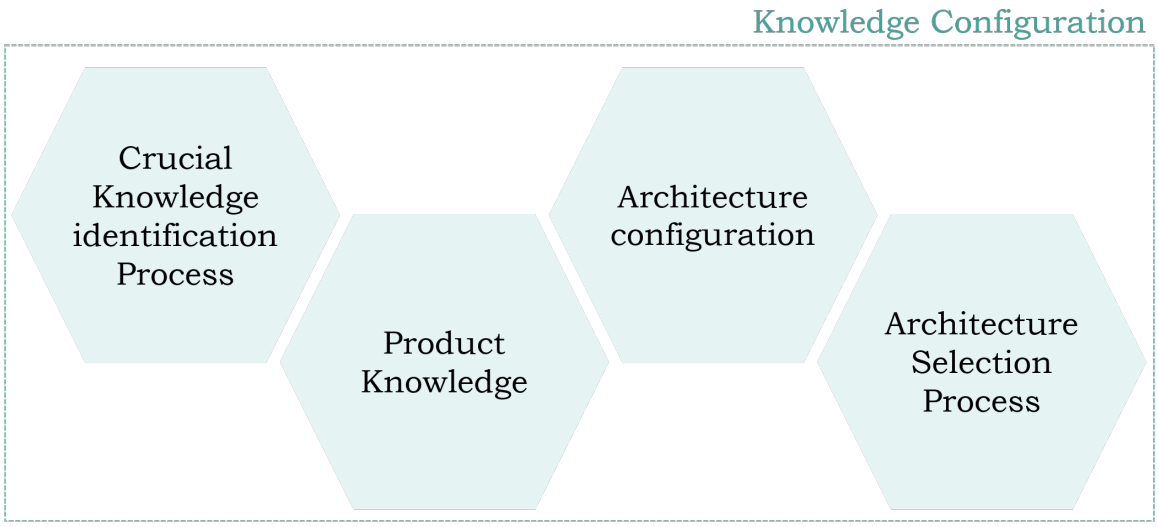


FIGURE 2.9 – Structure de KC dans CaTCoDD

2.7.7 Détail du concept *User Configuration (UC)*

La configuration d'utilisateur ou *User Configuration (UC)* est introduite pour définir les différents modèles métiers impliqués dans la collaboration. Cette classe contient les instances d'ICEs selon le besoin de chaque modèle métier (voir figure 2.10).

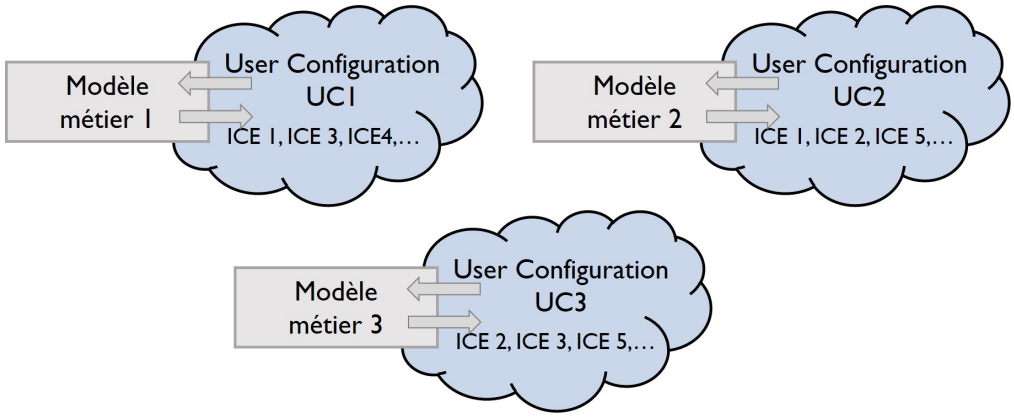


FIGURE 2.10 – Structure de UC dans CaTCoDD

2.7.8 Détail du concept *Product Knowledge (PK)*

Dans notre approche, nous avons gardé le même concept que celui introduit dans les travaux de Mcharek et al. [7]. Cette classe contient toutes les entités utilisées pour capitaliser les connaissances dans le processus de conception collaborative. Cependant, une nouvelle entité basée sur la théorie des catégories est utilisée pour sauvegarder les résultats.

2.7.9 Détail du concept *Architecture Configuration (AC)*

Configuration d'architecture ou *Architecture Configuration (AC)* est une classe créée afin de permettre l'évaluation de différentes architectures d'un système mécatronique. Cette classe regroupe la configuration de l'utilisateur et le processus de résolution des conflits. Grâce à cette nouvelle proposition, différentes architectures d'un système peuvent être comparées dès les premières phases de conception. Cela réduit considérablement les activités de correction et de ré-exécution. Pour chaque alternative une configuration d'architecture est créée. Cette configuration contiendra à son tour l'ensemble des configurations d'utilisateurs pour illustrer les différents modèles métiers utilisés par les ingénieurs. C'est au niveau de la configuration d'architectures que la gestion des conflits a lieu. L'AC est structurée comme représenté dans la figure (2.11).

2.7.10 Détail du concept *Conflict Resolution Process (CRP)*

Cette nouvelle classe est définie pour résoudre les conflits survenus entre les UCs durant la collaboration. Elle est basée sur le formalisme de la TC. Dans le modèle CDPPK, la résolution des conflits est basée sur un processus manuel sans tenir compte des dépendances entre les paramètres. Cependant, nous suggérons dans notre approche CaTCoDD, l'utilisation de la TC comme un cadre formel pour gérer les conflits tout en prenant en compte les relations entre les paramètres cruciaux. Deux cas se présentent dans notre approche : la gestion des conflits dans un seul modèle ou entre plusieurs modèles métiers. Un exemple d'un conflit entre deux valeurs différentes d'un paramètre est représenté par la figure (2.12). Deux modèles métiers sont considérés dans cet exemple. Le premier modèle revient à modéliser en 3D un piston dans un moteur à combustion interne et le deuxième représente un modèle de simulation Abaqus [6]. Chaque expert instancie l'ICE 1 qui contient le diamètre du piston. Chaque instance contient une valeur obtenue après les différentes activités de conception. Une incompatibilité entre les valeurs provoque un conflit entre les

6. Abaqus, Dassault Systèmes, `https://www.3ds.com/fr/produits-et-services/simulia/produits/abaqus/abaqusstandard/`

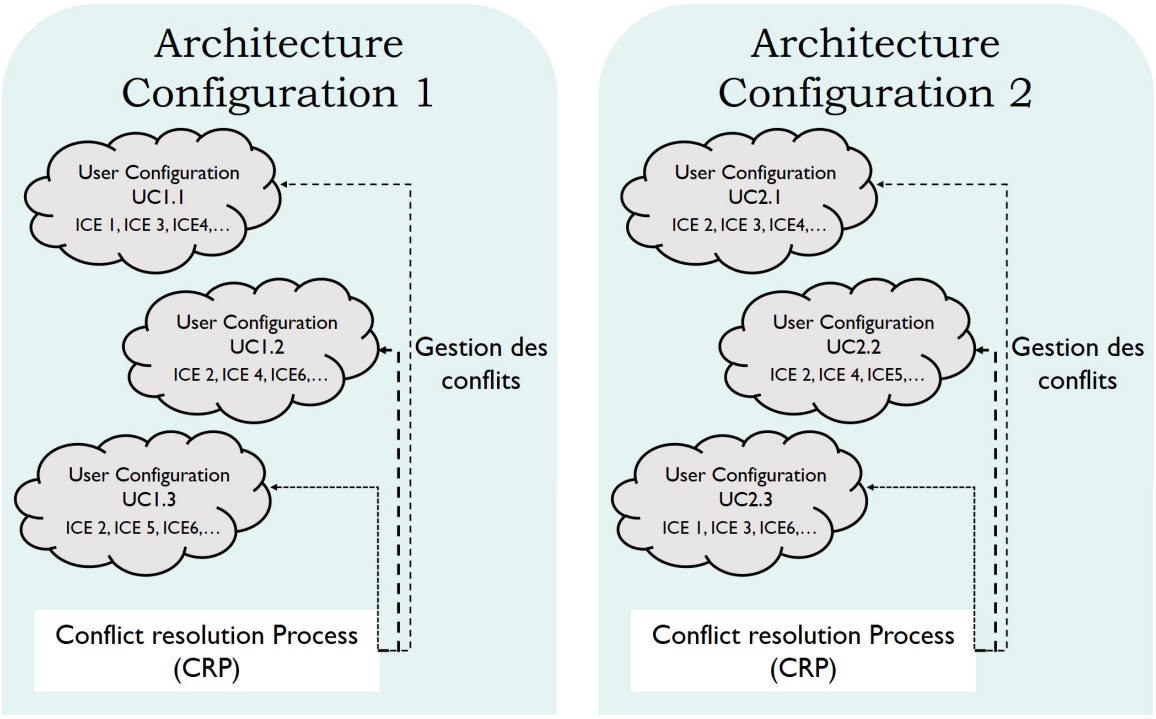


FIGURE 2.11 – Structure de la configuration d'architecture dans CaTCoDD

configurations d'utilisateurs.

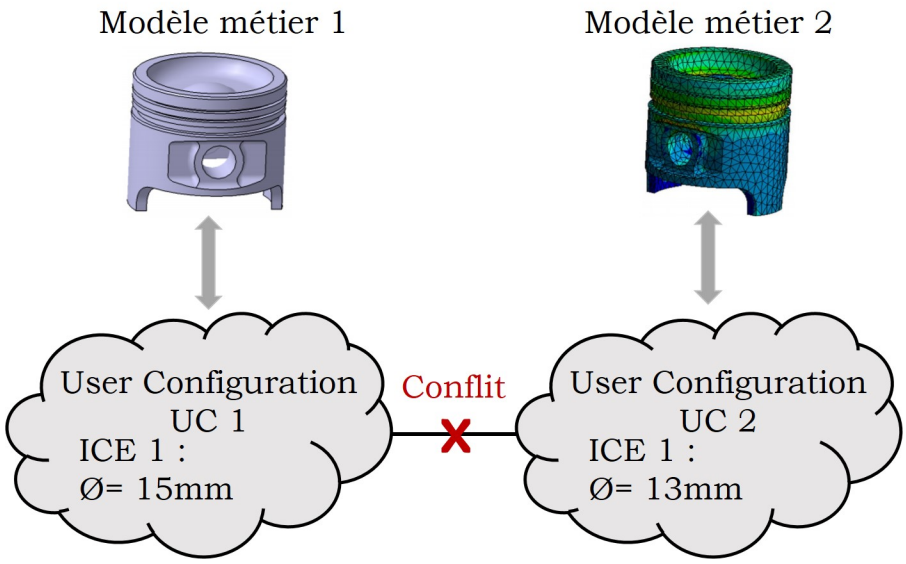


FIGURE 2.12 – Exemple d'un conflit entre deux configurations d'utilisateurs

2.7.11 Détail du concept *Architecture Selection Process (ASP)*

Après avoir créé toutes les architectures nécessaires, le processus de choix d'architectures peut commencer. Cette nouvelle classe se base sur la méthode AHP pour comparer les architectures et calculer les scores de chaque alternative. Un certain nombre de critères est défini au début de ce processus. En se basant sur ces critères, les valeurs obtenues à la fin de la collaboration et la gestion des conflits seront utilisées pour calculer le score de chaque architecture. L'architecture avec le score le plus élevé sera considérée comme l'architecture qui répond le mieux aux critères définis précédemment. Cette classe sera plus détaillée dans les chapitres suivants.

2.8 Méta-Modèle de CaTCoDD

Dans cette section, une formalisation de la méthodologie CaTCoDD est effectuée sous forme d'un méta-modèle que nous désignons par *CaTCoDD model*. Ce modèle s'appuie sur le langage orienté objets UML et sur le principe des approches d'ingénierie dirigée par les modèles (ou *Model-Driven Engineering* (MDE)). Les approches MDE permettent de représenter un concept indépendamment des langages et des plateformes.

Un modèle est défini selon Fisher et al. [112] comme étant une abstraction significative et pertinente d'un système. Ce modèle est particulièrement important pour l'ingénieur qui doit analyser, spécifier, concevoir et vérifier les systèmes, ainsi que partager des informations avec d'autres parties prenantes. Pour définir un tel modèle, la mise en oeuvre d'un méta-modèle est nécessaire. Ce méta-modèle fournit le langage requis pour établir un modèle dans un contexte particulier. A partir d'un méta-modèle, il est possible d'obtenir différents modèles d'applications. Chaque modèle fait référence à un projet de collaboration particulier. En d'autres termes, ces modèles d'applications ne sont qu'une instanciation du méta-modèle générique comme reporté sur la figure (2.13). Ces modèles doivent être conformes au méta-modèle CaTCoDD afin qu'ils soient réutilisables et utiles.

Nous avons choisi d'établir le méta-modèle CaTCoDD en se basant sur le langage UML, en tant qu'un langage de modélisation orienté objet. Notre méta-modèle est constitué de quatre parties principales (voir figure 2.9). La première partie est réservée à l'identification des connaissances cruciales. La deuxième partie fait référence à la capitalisation des connaissances cruciales en se basant sur le concept d'ICE. La partie suivante supporte la génération de plusieurs architectures d'un système mécatronique en gérant les conflits apparus après le partage des connaissances. La comparaison et le choix d'architectures représentent la dernière partie du méta-modèle. Enfin, les différentes parties sont reliées les unes aux autres afin d'assurer la cohérence de notre méta-modèle.

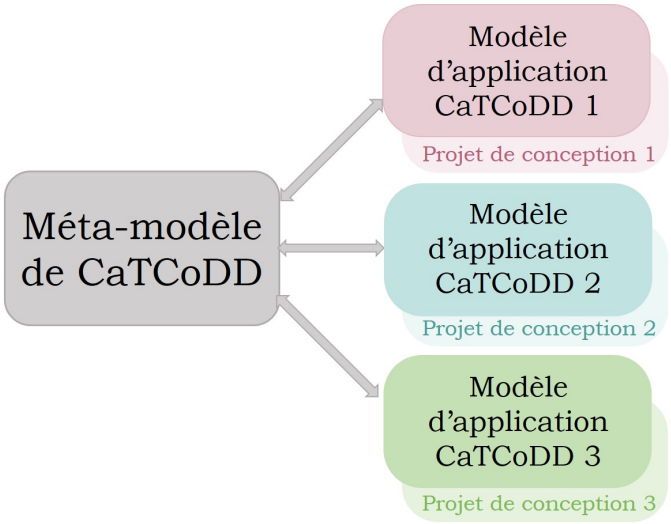


FIGURE 2.13 – Définition des projets de collaboration en se basant sur un méta-modèle

Dans ce nouveau modèle, chaque architecture du système est considérée comme une configuration d'architecture (AC). Pour chaque architecture, plusieurs domaines d'expertise sont considérés. Ces domaines sont représentés par une configuration d'utilisateur (UC). Le méta-modèle de CaTCoDD est représenté par la figure (2.14). En se référant à la norme ISO 10007 : 2017 [40], le but de la gestion de configuration consiste à réduire les phases de re-conception en permettant aux équipes de conception de travailler sur une définition cohérente du produit. Une configuration est définie comme un ensemble de caractéristiques techniques et fonctionnelles d'un produit. Ce concept est représenté par la configuration des connaissances (KC) dans notre méta-modèle CaTCoDD.

La classe CKIP est composée de deux packages : *Parameter Category (PC)* et *classify Knowledge* comme le montre la figure (2.14). Ces packages définissent les concepts pour identifier les connaissances cruciales. La sous-classe PC est une représentation de connaissances en se basant sur la TC, où chaque paramètre utilisé dans les modèles métiers est traduit sous forme d'une catégorie contenant un ensemble d'objets (différentes instances du paramètre établies par différents modèles métiers) reliés entre eux à l'aide des morphismes (pour illustrer les activités d'ingénierie). La deuxième sous-classe dans CKIP met en valeur une classification des paramètres, où les paramètres représentés avec des catégories contenant plusieurs objets, sont considérés cruciaux. Ces paramètres sont regroupés dans la classe nommée $KClass_1$, un acronyme qui fait référence à *Knowledge Classification*.

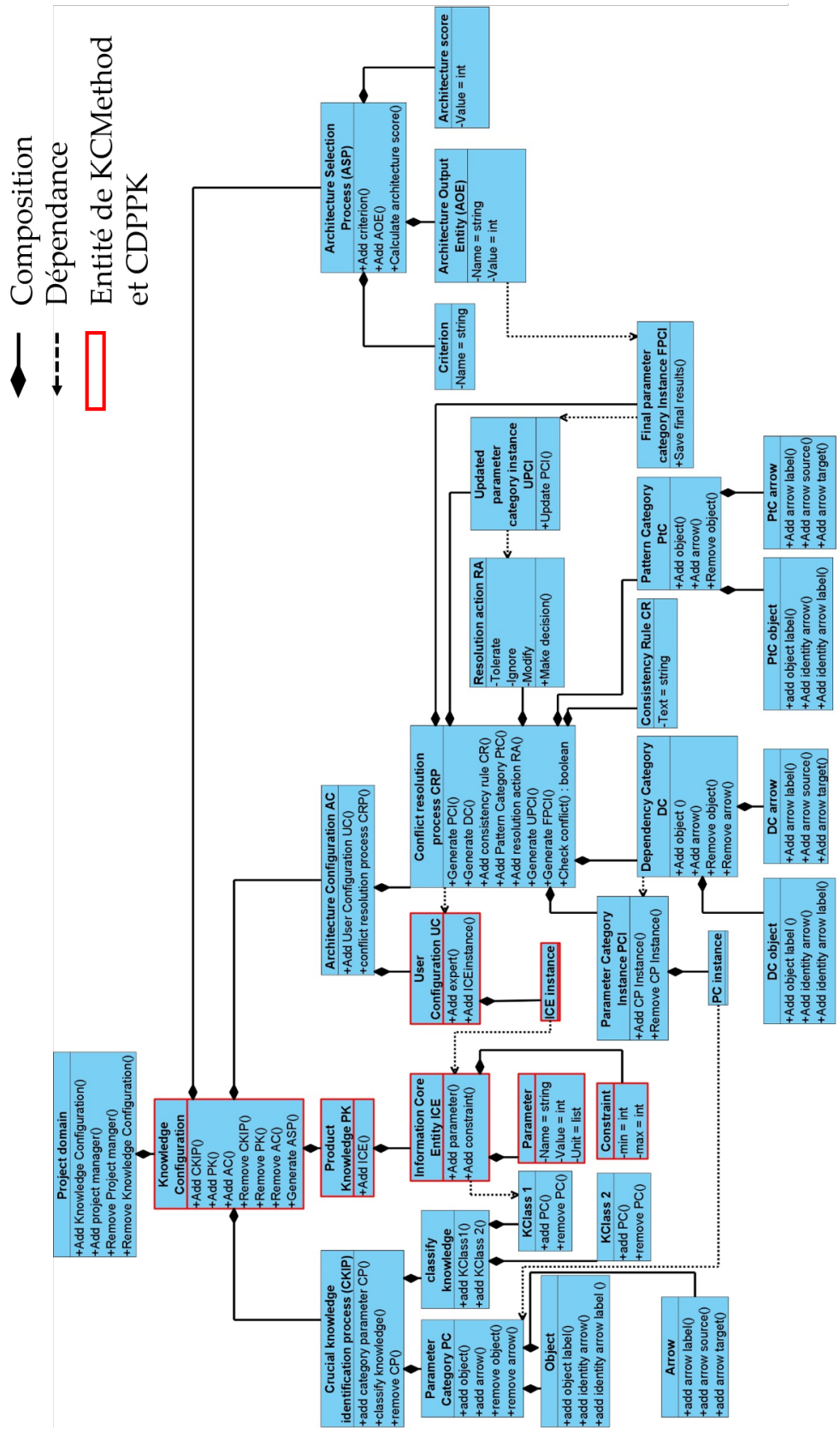


FIGURE 2.14 – Méta-modèle CaTCoDD

Cependant, ceux qui contiennent un seul objet ne seront pas cruciaux à la collaboration et ne nécessitent pas la capitalisation et le partage. Ces paramètres sont regroupés dans $KClass_2$.

La classe PK regroupe toutes les ICEs nécessaires à la collaboration. Les ICEs permettent de capitaliser uniquement les connaissances cruciales (regroupées dans $KClass_1$). Chaque ICE est constituée d'un ensemble de paramètres et contraintes.

Le méta-modèle CaTCoDD contient une autre classe nommée *Architecture Configuration AC*. Cette classe contient à son tour les configurations d'utilisateurs et le processus de gestion des conflits ou *Conflict Resolution Process (CRP)*. Cette classe se repose sur la théorie des catégories qui aide à la fois, à formaliser les modèles hétérogènes, faciliter la détection des conflits via les constructeurs de la TC et assurer la traçabilité.

La dernière classe du méta-modèle CaTCoDD est appelée *Architecture Selection Process (ASP)*. C'est au niveau de cette classe que la comparaison entre les différentes architectures du système a lieu. La classe ASP regroupe à son tour trois sous-classes. La première sous-classe englobe les différents critères de comparaison et la deuxième nommée *Architecture Output Entity (AOE)* rassemble les valeurs obtenues après la collaboration et qui seront utiles à la comparaison. La dernière entité de la classe ASP contient les scores de chaque architecture qui sont obtenus en se basant sur la méthode AHP.

Après avoir décrit notre approche CaTCoDD et son méta-modèle associé, il est important de souligner l'apport de notre proposition par rapport aux modèles présentés dans la littérature. CaTCoDD est un modèle de connaissance qui assure une gestion des données de granularité fine pour faciliter la collaboration entre les modèles métiers impliqués dans la conception. Dans notre modèle, on a recours aux ICEs pour capitaliser les connaissances qui sont ensuite contextualisées pour être employées dans un contexte donné. Par ailleurs, CaTCoDD permet d'identifier d'une manière formelle les connaissances utiles à la collaboration. Cette étape a été négligée dans les modèles existants (CDPPK, KCMethod, etc.). Nous proposons ainsi une méthode formelle basée sur la théorie des catégories pour extraire les connaissances cruciales. En se basant sur la même théorie, les conflits entre les modèles métiers sont détectés et gérés. Enrichi par la méthode AHP, le modèle CaTCoDD permet de comparer différentes architectures d'un système mécatronique pour aider le chef de projet à choisir l'architecture qui permet d'avoir un compromis entre tous les critères. De ce fait, grâce au modèle CaTCoDD les limites présentées dans les modèles existants dans la littérature ont été soulevées comme le montre le tableau (2.1).

TABLE 2.1 – Apport du modèle CaTCoDD par rapport aux modèles existants dans la littérature

Critères	Identification des connaissances	Capitalisation des connaissances	Gestion des conflits	Choix d'architectures
E2KS	Non	KPac	Non	Non
COLIBRI	Non	Contraintes	Non	Non
KCMethod	Manuelle	ICE	Partielle	Non
CDPPK	Manuelle	ICE décomposable	Partielle	Non
CaTCoDD	Formelle avec la TC	ICE décomposable	Totale avec la TC	Oui

2.9 Développement d'un démonstrateur pour la validation

Pour illustrer la capacité de la méthodologie CaTCoDD, un démonstrateur avec Matlab GUI (*Graphical User Interface*) a été implémenté. Le démonstrateur, basé sur la méthodologie CaTCoDD et son méta-modèle, est dédié à la conception collaborative des systèmes mécatroniques. Cet outil permet d'assurer une collaboration et un échange dynamique entre les modèles métiers participants à la collaboration. L'objectif de ce démonstrateur est d'illustrer les différentes étapes de la méthodologie CaTCoDD et de valider également ses aspects en se basant sur une implémentation réelle. L'architecture envisagée pour notre démonstrateur est constituée de quatre niveau interdépendants comme représenté par la figure (2.15). Le premier niveau fournit un espace où les connaissances cruciales peuvent être extraites pour faire ensuite l'objet de la capitalisation. Ce niveau est représenté par la fenêtre *Crucial Knowledge Identification Process (CKIP)*. Le deuxième niveau, illustré par la fenêtre *Product Knowledge (PK)*, sert à capitaliser les connaissances sous forme d'ICEs pour des perspectives de réutilisation dans des projets de conception ultérieurs. Une troisième fenêtre dans notre démonstrateur fournit un espace de collaboration entre tous les ingénieurs disciplinaires où il est possible de partager les connaissances pour assurer la cohérence du système. Cette fenêtre permet également de gérer les conflits apparus après l'échange entre les ingénieurs disciplinaires. Le dernier niveau du démonstrateur permet au chef de projet de comparer les architectures du système afin de choisir l'architecture qui satisfait au mieux les critères.

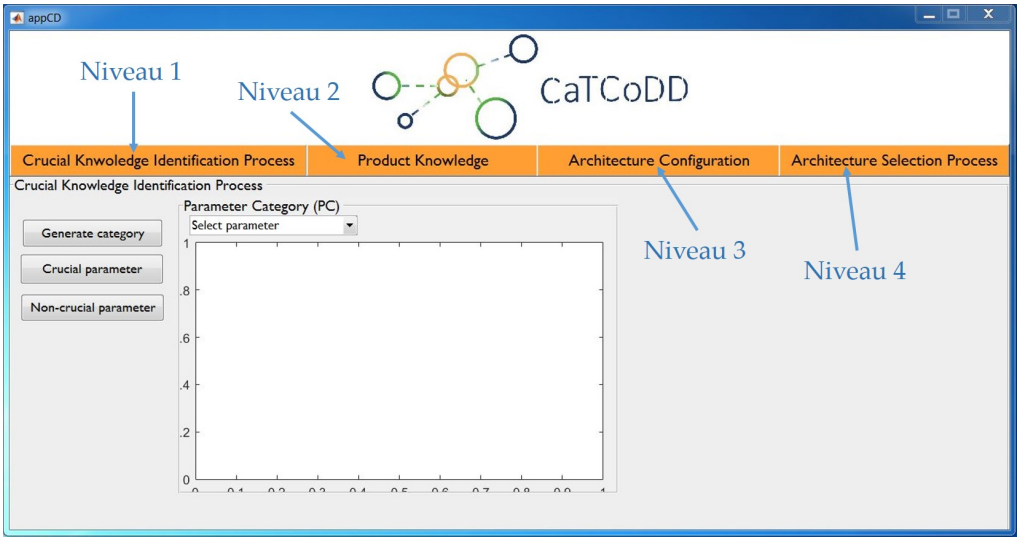


FIGURE 2.15 – Les différents niveaux dans le démonstrateur CaTCoDD avec Matlab GUI

2.9.1 Crucial Knowledge identification Process (CKIP)

C'est la première partie du démonstrateur où les connaissances cruciales sont identifiées en se basant sur la TC. La première étape dans ce processus consiste à créer des catégories représentant les paramètres utilisés par les ingénieurs disciplinaires (*Parameter Category PC*). Chaque catégorie regroupe les différentes instances du paramètre provenant de différents modèles métiers. Ces instances sont les objets de la catégorie qui sont reliés entre eux à l'aide des morphismes. A titre d'exemple, la figure (2.16) présente une catégorie créée dans le démonstrateur. Le paramètre représenté par cette catégorie est une puissance maximale d'un moteur (P_m). Ce paramètre est défini dans un premier temps, par le premier modèle métier dédié à la spécification des exigences (EM_r). Dans un second temps, et après des simulations multi-physiques, la puissance maximale est calculée dans le deuxième modèle métier consacré à la modélisation multiphysique (EM_{mp}). Enfin, en se basant sur une activité de choix des composants la puissance maximale est choisie dans le troisième modèle métier (EM_{COTS}). Le passage d'un modèle métier à un autre est défini sous forme d'un morphisme (flèche) comme le montre la figure (2.16). Pour représenter l'évolution interne de chaque objet (ou instance d'un paramètre), un morphisme d'identité est défini.

Une fois des catégories créées pour chaque paramètre, on peut passer à l'étape d'extraction des connaissances cruciales. Le paramètre dont la catégorie contient au moins deux objets est considéré crucial. Une liste des paramètres cruciaux et non cruciaux est présentée dans le démonstrateur (Voir figure 2.17).

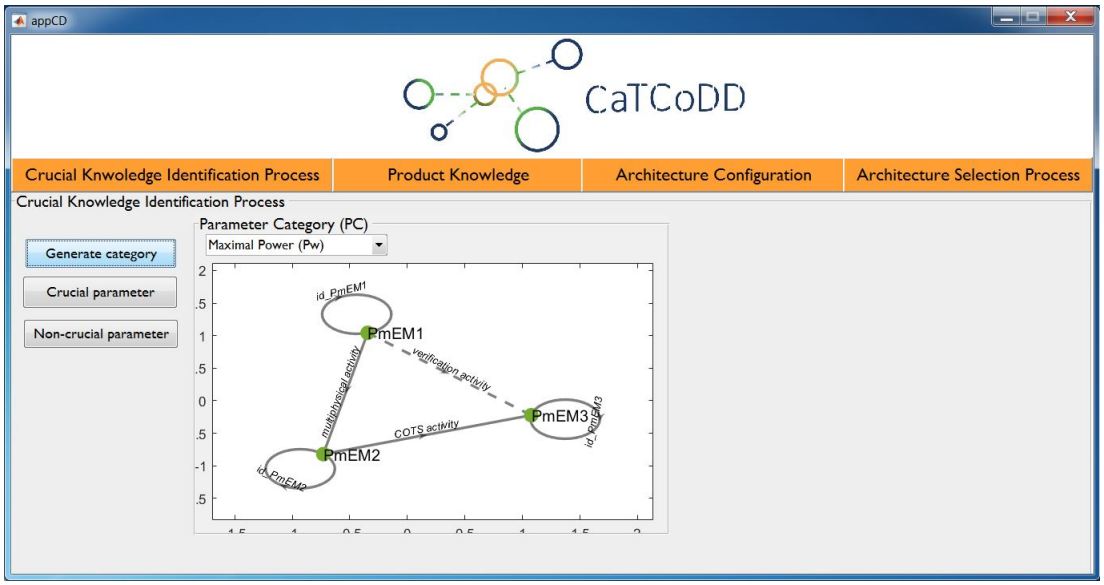


FIGURE 2.16 – Identification de connaissances cruciales avec le démonstrateur CaTCoDD

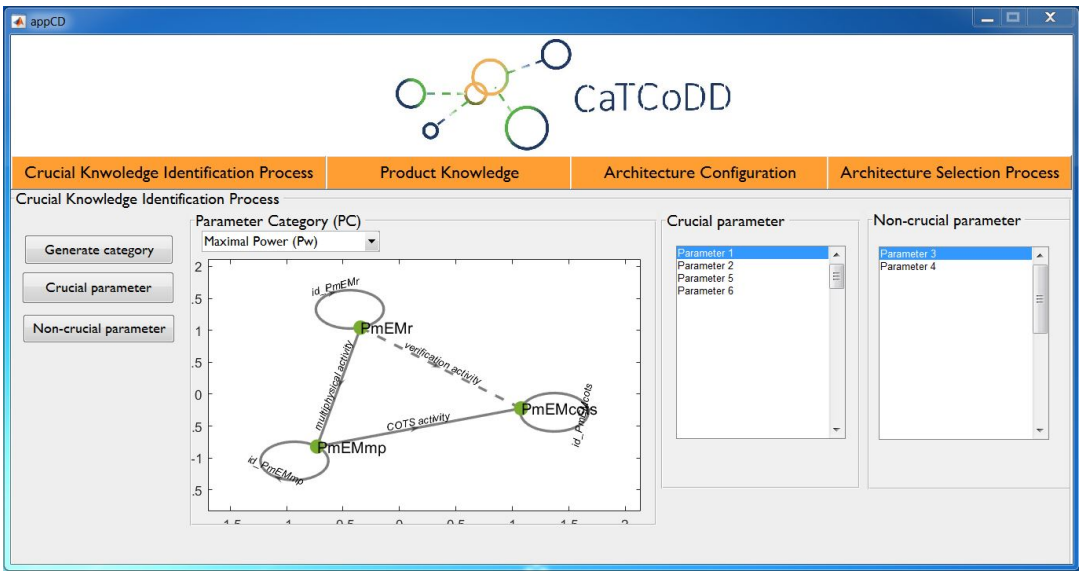


FIGURE 2.17 – Liste des paramètres cruciaux et non cruciaux dans CaTCoDD

2.9.2 Product Knowledge (PK)

Après avoir identifié les paramètres cruciaux, leur capitalisation devient nécessaire. En ce sens, un ensemble d'ICEs est créé. Ces ICEs seront ensuite instanciées par les ingénieurs disciplinaires afin de les enrichir par des valeurs obtenues après les différentes activités d'ingénierie (voir figure 2.18).

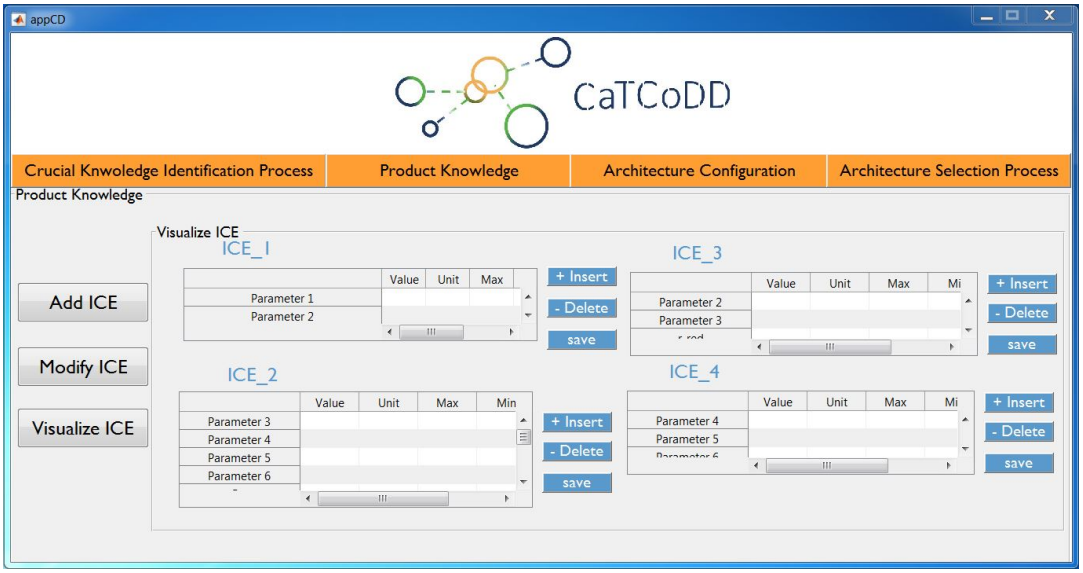


FIGURE 2.18 – ICEs créées dans CaTCoDD

2.9.3 Architecture Configuration (AC)

Le troisième niveau du démonstrateur s'intéresse à la collaboration entre les ingénieurs disciplinaires. C'est à ce niveau que les différentes architectures sont représentées par des configurations d'architectures (ACs). Chaque AC contiendra un ensemble de configurations d'utilisateurs (UCs) relatifs aux modèles métiers utilisés par les ingénieurs. Chaque ingénieur disciplinaire instancie les ICEs dont il a besoin. A ce niveau des valeurs sont attribuées à chaque paramètre selon les résultats obtenus depuis chaque modèle métier. Un exemple d'UCs créées dans le démonstrateur est reporté sur la figure (2.19).

Une fois les UCs créées, les ingénieurs disciplinaires peuvent les publier afin de vérifier l'apparition de quelques conflits. Pour détecter les conflits, le concept de la TC est utilisé pour formaliser les modèles métiers et faciliter ainsi la localisation des incohérences. En premier lieu, les catégories des paramètres créées précédemment sont mises à jour et enrichies par des valeurs pour obtenir une instance de la catégorie de paramètre ou *Parameter Category Instance* (*PCI*). Il est important de souligner

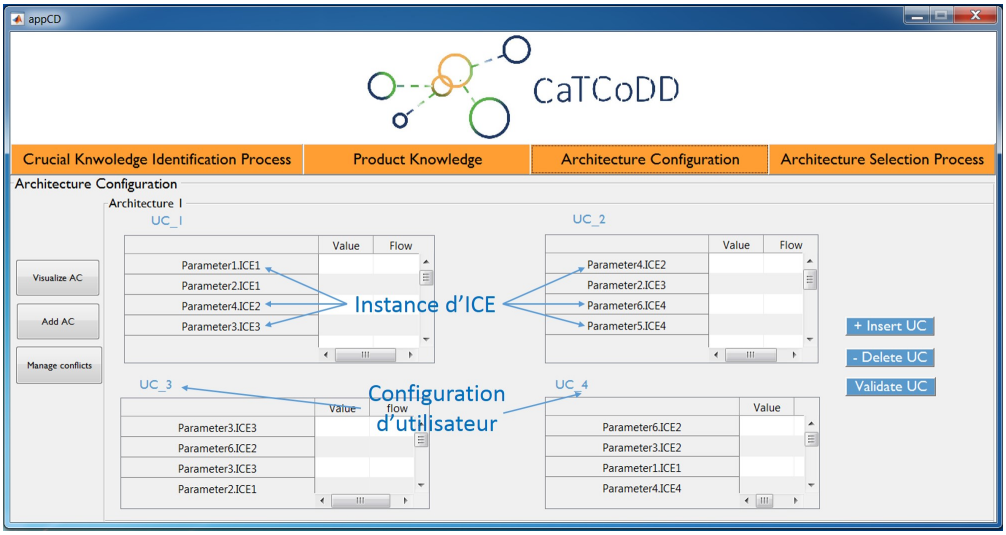


FIGURE 2.19 – UCs créées dans CaTCoDD

que les paramètres cruciaux sont dépendants les uns aux autres. Pour cela, une catégorie de dépendance ou *Dependancy Category* (DC) est créée afin de mettre en valeur les liens de dépendance entre les paramètres. Les objets de cette catégorie représentent les paramètres et les morphismes illustrent les liens entre eux (voir figure 2.20).

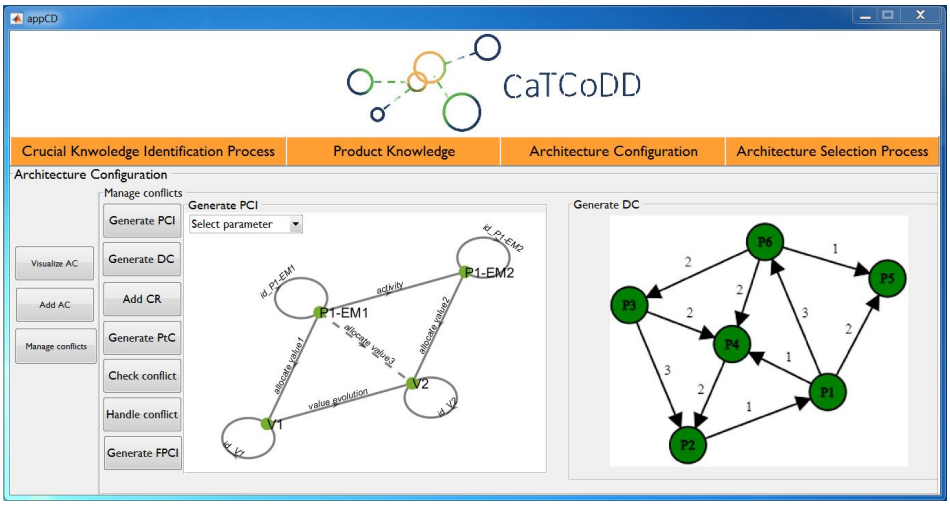


FIGURE 2.20 – Instance de catégorie du paramètre (PCI) et catégorie de dépendance (DC) dans CaTCoDD

En deuxième lieu, des règles de cohérences pour mettre en valeur les contraintes à respecter sont définies. Ces règles sont ensuite transformées en *pattern* pour préciser les conflits qui peuvent apparaître entre les différentes valeurs d'un paramètre (Voir

figure 2.21).

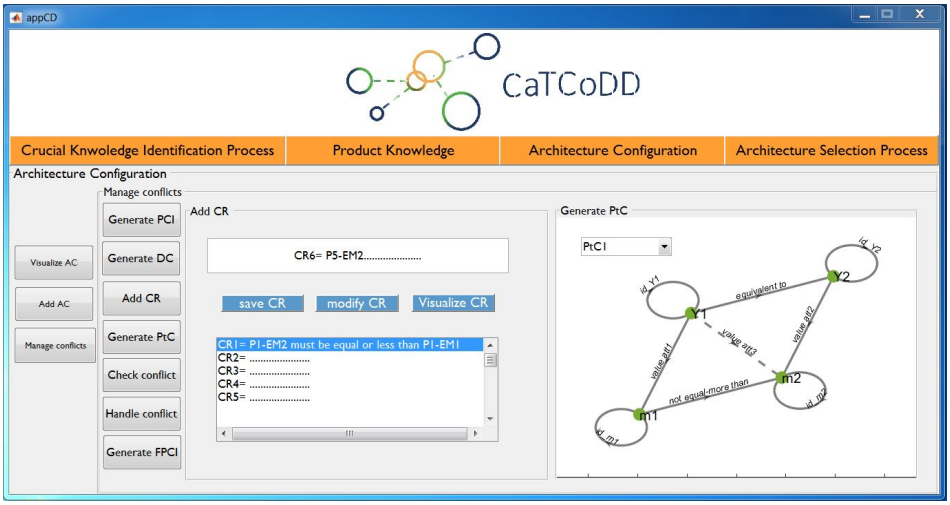


FIGURE 2.21 – Règles de cohérences et catégorie de pattern dans CaTCoDD

En dernier lieu, le chef de projet peut détecter les conflits à travers un mapping entre les catégories des paramètres et des patterns. Ce mapping s'effectue d'une manière formelle en appliquant un foncteur entre les catégories des paramètres et les catégories des patterns. La méthode de localisation des conflits sera plus détaillée dans la suite. Une fois les conflits détectés, le chef projet commence la gestion des incohérences. Afin de gérer les conflits, trois actions peuvent avoir lieu : une action de tolérance, d'ignorance ou de résolution où un échange étroit entre les ingénieurs disciplinaires est établi afin de trouver un compromis entre eux.

Après chaque action de résolution, une mise à jour des $PCIs$ est établie afin de tenir en compte les modifications effectuées. Cette mise à jour (*Updated Parameter Category Instance* ($UPCI$)) est suivie d'un mapping entre les DCs et les $PCIs$ pour détecter de nouveau les conflits. Ces dernières étapes se répètent jusqu'à la résolution de l'ensemble des conflits. Les valeurs finales seront enregistrées dans les instances finales des catégories des paramètres ou *Final Parameter Category Instance* $FPCI$.

2.9.4 Architecture Selection Process (ASP)

Le dernier niveau de CaTCoDD consiste à évaluer les différentes architectures du système mécatronique en s'appuyant sur la méthode AHP. Il suffit de préciser les critères de choix et extraire les valeurs correspondantes pour effectuer la comparaison comme reporté sur la figure (2.22).

À ce niveau, le chef de projet dispose de toutes les connaissances nécessaires pour choisir la meilleure architecture. De ce fait, l'ingénieur système commence

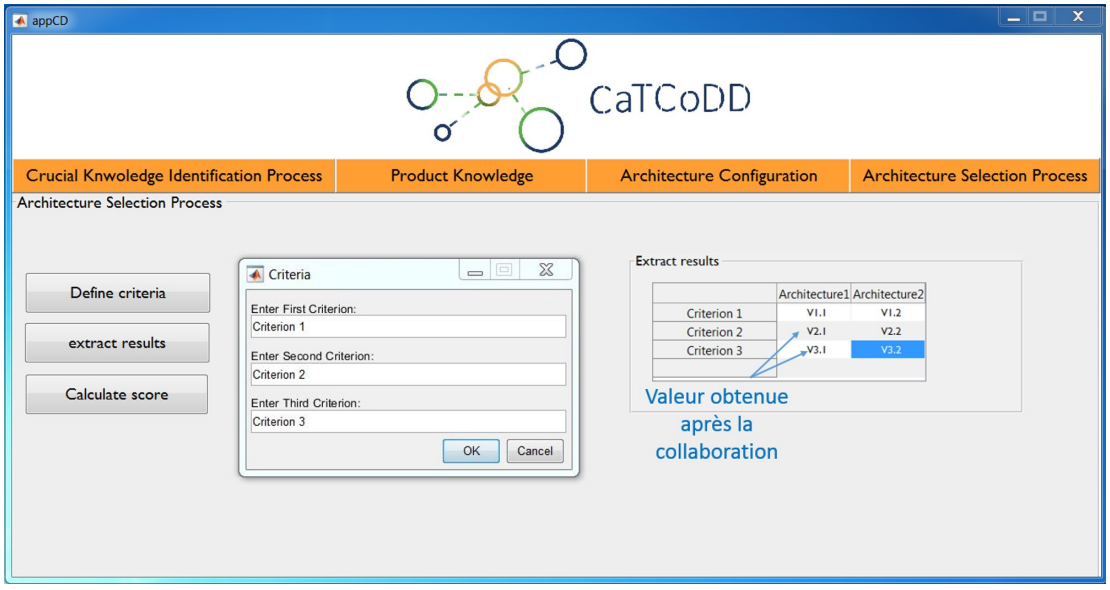


FIGURE 2.22 – Définition des critères pour le choix d'architectures

par définir la matrice de comparaison par paire des critères. Comme dans tout processus de décision, les critères n'ont pas la même importance. Par conséquent, il est essentiel de définir les priorités des critères. Ces poids sont basés sur une échelle de comparaison numérique (voir tableau 2.2) proposée par Saaty [91]. La définition de l'importance d'un critère par rapport à un autre est établie dans le démonstrateur CaTCoDD comme nous pouvons le voir sur la figure (2.23).

TABLE 2.2 – Échelle de comparaison proposée par Saaty [91]

Jugement	**Valeur numérique**
extrêmement important	9 8
Très fortement plus important	7 6
fortement plus important	5 4
Modérément plus important	3 2
équivalent	1

Après avoir créé la matrice de comparaison par paires il est important d'établir pour chaque critère de décision, une matrice qui permet de comparer les différentes alternatives. Cette matrice est remplie en se basant sur la même échelle considérée

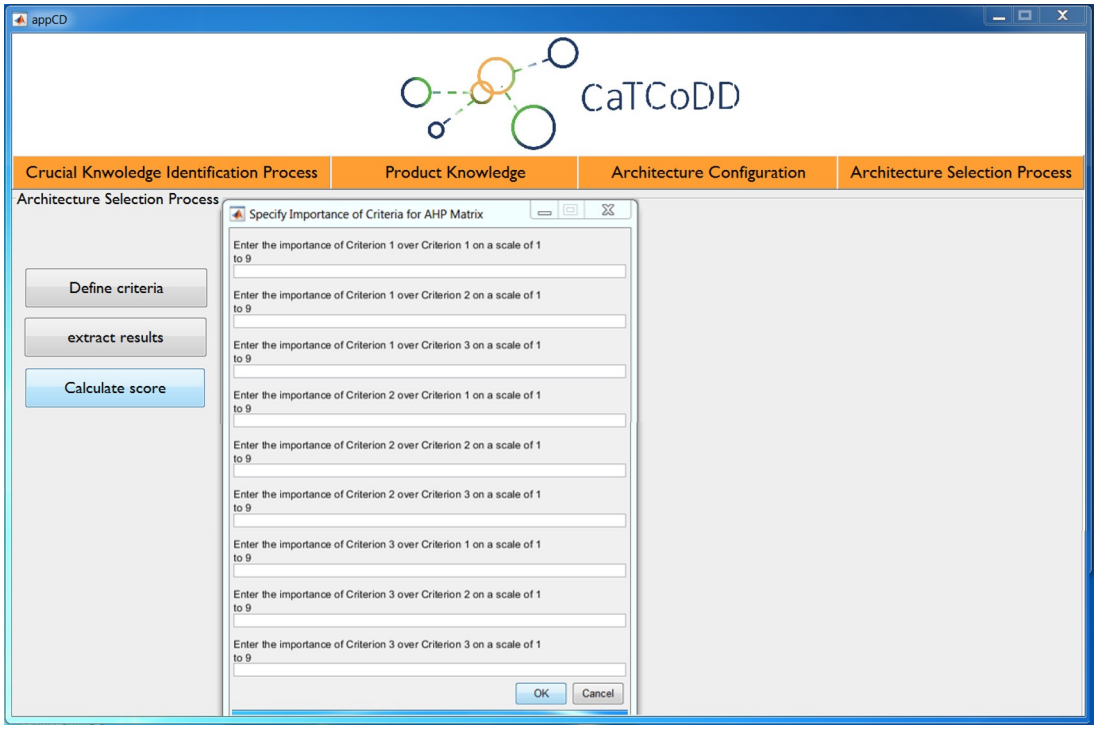


FIGURE 2.23 – Matrice de comparaison par paires des critères dans CaTCoDD

dans la création de la matrice de comparaison par paires (Voir figure 2.24). Enfin, les scores de chaque alternative sont calculés. L'alternative qui possède le score le plus élevé représente la solution qui répond le mieux aux critères de choix. Les matrices créées au cours de cette étape ainsi que le calcul de score pour les différentes architectures seront détaillés de plus dans le chapitre 3. Plus de détails sur la programmation de la méthode ASP en utilisant Matlab sont fournis dans l'annexe 1.

Grâce au milieu collaboratif CaTCoDD, un échange entre les parties prenantes du projet de conception devient possible où le chef de projet interagit avec l'ingénieur système ainsi que les ingénieurs disciplinaires. Quatre niveaux dans le démonstrateur CaTCoDD ont été présentés dans ce paragraphe comme indiqué dans la figure (2.25). Une vue plus réelle de la méthodologie CaTCoDD est obtenue avec l'implémentation du démonstrateur CaTCoDD. L'identification des connaissances cruciales a lieu au premier niveau en se basant sur la théorie des catégories. Les connaissances cruciales sont ensuite capitalisées dans le deuxième niveau du démonstrateur. Cette étape est suivie de la collaboration et l'échange entre les ingénieurs disciplinaires. Les conflits sont également localisés et gérés à ce niveau. La dernière partie permet de supporter la comparaison de différentes architectures d'un système mécatronique.

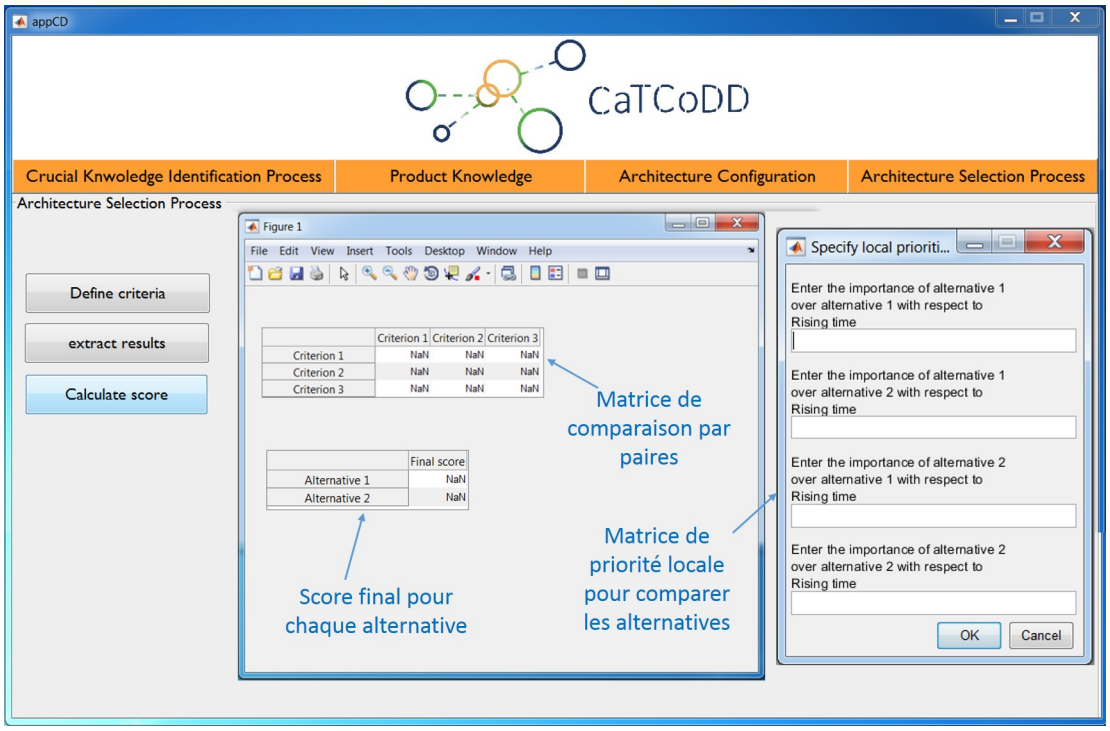


FIGURE 2.24 – Matrice de priorité locale et les scores finaux de chaque architecture

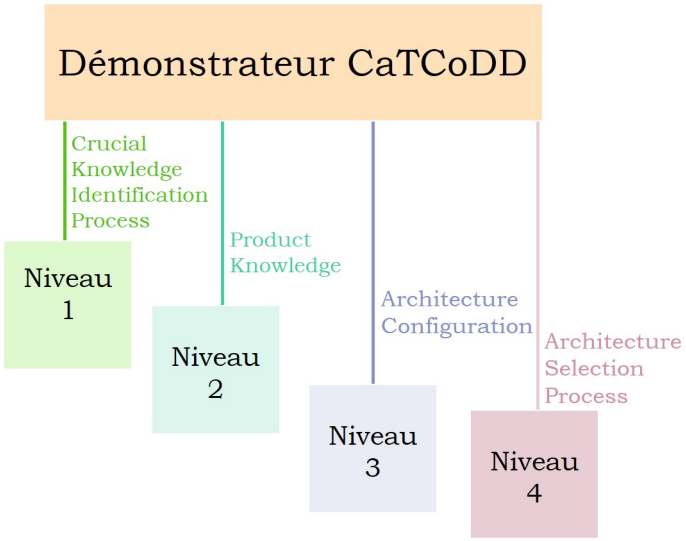


FIGURE 2.25 – Architecture du démonstrateur CaTCoDD

2.10 Conclusion

Nous avons commencé ce chapitre par la définition du triplet donnée-information-connaissance afin de mettre l'accent sur le fait qu'elles sont différentes mais dépendantes. Le concept de la gestion des connaissances ou *knowledge Management* a été également présenté dans ce chapitre. Par la suite, nous avons présenté une nouvelle méthodologie basée sur la théorie des catégories pour la conception collaborative et l'aide à la décision pour les systèmes mécatroniques. Nous avons désigné cette méthode par CaTCoDD (*Category Theory-based Collaborative Design and Decision-making*). Trois étapes principales constituent cette approche : l'initialisation, la collaboration et l'aide à la décision. Nous avons montré l'efficacité de la méthodologie CaTCoDD, à travers une comparaison entre les méthodologies de collaboration déjà existantes et notre approche. Un méta-modèle en utilisant le langage UML a été réalisé dans ce chapitre. Ce méta-modèle ainsi que ses différentes classes se basent sur les concepts fondamentaux de la méthodologie CaTCoDD. Enfin, afin de permettre une validation pertinente et une vision plus réelle de notre approche, un démonstrateur a été développé. Ce démonstrateur implémente l'identification des connaissances cruciales, la collaboration, la gestion des conflits et l'aide à la décision. Les méthodes développées pour identifier les connaissances cruciales, pour gérer les conflits en se basant sur la théorie des catégories et pour le choix d'architectures feront l'objet du chapitre suivant.

Chapitre 3

Méthodologies pour l'identification des connaissances cruciales, la gestion des conflits et le choix d'architectures

Dans ce chapitre, on va se focaliser sur l'identification des connaissances cruciales et la gestion des conflits en se basant sur la théorie des catégories. Le choix d'architectures sera également détaillée. La première partie de ce chapitre sera consacrée à la méthodologie proposée pour identifier d'une manière formelle les connaissances nécessitant la capitalisation et le partage. La deuxième partie est dédiée à la méthodologie de gestion des conflits où les conflits apparus après la collaboration et le partage des connaissances sont localisés et gérés. La troisième partie concernera la méthodologie développée afin de choisir l'architecture la plus appropriée. Ces trois approches correspondent respectivement au deuxième, troisième et quatrième axes de recherche présentés précédemment.

3.1 La méthodologie Crucial knowledge Identification Process (CKIP)

Comme on l'a souligné précédemment, avec la complexité croissante ainsi que la variété des domaines dans la conception mécatronique, il est important d'adopter une collaboration multidisciplinaire dans le processus de conception [113]. Cette collaboration implique l'intégration d'activités multidisciplinaires où chaque ingénieur a recours à un ensemble de connaissances spécifiques à son domaine d'expertise. De ce fait, déterminer les connaissances qui doivent faire l'objet de capitalisation dans le processus de conception est une tâche difficile et chronophage. Pour cela, nous proposons une nouvelle méthodologie formelle basée sur la théorie des catégories (TC) pour identifier les connaissances cruciales. La motivation de l'utilisation de cette théorie dans notre méthodologie est que la TC est considérée comme un outil puissant et une méthode de modélisation formelle pour capturer les interactions entre des composants hétérogènes de manière naturelle [114]. En outre, cette théorie mathématique se concentre sur les relations entre les objets au lieu d'étudier leur représentation, ce qui est adapté à la conception collaborative, où le partage et l'échange des connaissances entre les parties prenantes est un concept de premier ordre [115]. De plus, les différentes activités d'ingénierie sont liées à des modèles métiers spécifiques. Ces modèles sont composés d'un ensemble d'éléments interconnectés et peuvent être représentés sous forme de graphes [112]. Ces graphes ont une signification formelle dans la TC et portent toutes les intuitions qui proviennent de la pratique. Ainsi, la TC est considérée comme un outil puissant permettant de fournir un cadre unifié et formel pour illustrer les interconnexions entre les différentes parties prenantes. L'approche que nous proposons, illustrée dans la figure (3.1), est composée de deux phases. La première est dédiée à la formalisation des différents modèles impliqués dans le processus de collaboration et la deuxième concerne l'extraction des connaissances cruciales.

3.2 Architecture de la méthodologie CKIP

Sur la base du processus présenté dans la section précédente, nous définissons le problème d'identification des connaissances cruciales comme un ensemble de 3 éléments : CKIP = <Expert Model (EM), Parameter Category (PC), Knowledge Classification (KClass)>.

— les modèles métiers ou *Expert Models* (EM) font référence à un ensemble de modèles, qui sont relatifs à différentes activités de conception et sont impliqués dans le processus de collaboration (EM_1, EM_2, ..., EM_n). Ces modèles métiers peuvent prendre différentes formes telles que : un modèle

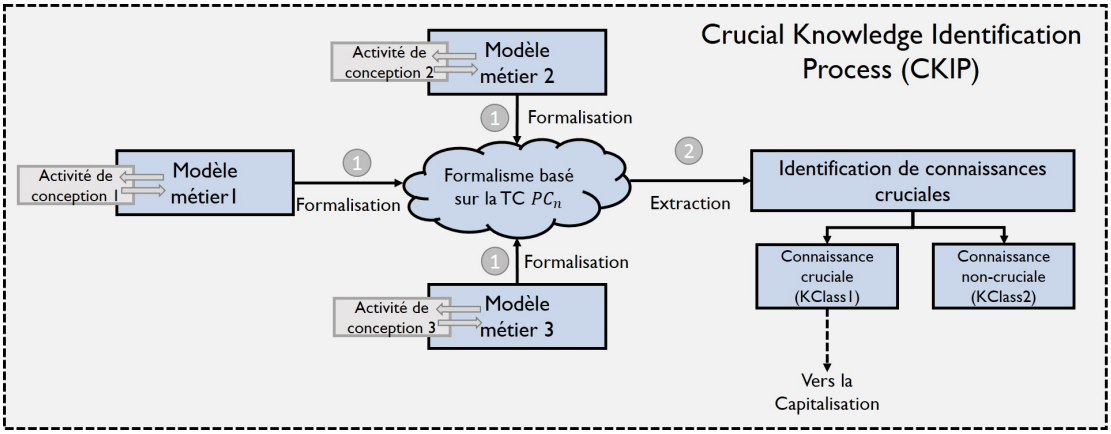


FIGURE 3.1 – Structure de la méthodologie CKIP pour l'extraction des connaissances cruciales

multi-physique utilisant le langage Modelica dans l'environnement Dymola, un modèle de contrôle avec le logiciel Simulink, un modèle COTS pour sélectionner les composants appropriés avec les propriétés obtenues par la simulation, un modèle 3D avec l'environnement CATIA pour vérifier l'intégration de l'ensemble du mécanisme, etc.

— Catégorie des paramètres ou *Parameter Category* (PC) représente les graphes unifiés basés sur la théorie des catégories, (PC_1,PC_2,..,PC_m), ce formalisme commun contient tous les paramètres utilisés dans les différents modèles métiers et permet d'identifier ceux qui sont cruciaux et qui nécessitent la capitalisation (voir figure 3.2). Un ensemble de cinq éléments PC = <O, id, A_{rr}, Lo, L_{Arr}> est un graphe dirigé. Dans le contexte de la théorie des catégories, ce graphe représente une catégorie où O = (O_1,...,O_n) est un ensemble d'objets. Chaque objet a sa propre identité (id). Ces objets sont les différentes instances d'un paramètre utilisé par les modèles métiers. Les objets sont reliés entre eux par un ensemble de flèches ou de morphismes A_{rr}. Les objets et flèches ont des attributs L_o et L_{Arr} respectivement. Un exemple de PC, créé selon la définition précédente, est illustré par la figure (3.2).

— La classification des connaissances ou *Knowledge Classification* (KClass). Une connaissance est considérée comme cruciale si sa catégorie (PC) n'est pas une catégorie discrète. Cela signifie qu'il n'y a pas seulement des flèches d'identité, mais qu'il existe aussi différents morphismes entre les objets, ce qui représente les différentes activités d'ingénierie. La $KClass_1$ contient tous les paramètres cruciaux qui seront capitalisés au cours du processus de collaboration. Néanmoins, la $KClass_2$ regroupe les paramètres non-cruciaux.

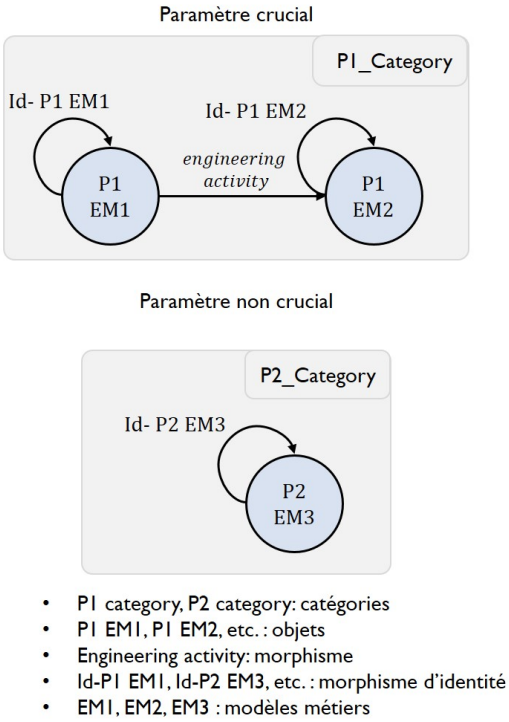


FIGURE 3.2 – Exemple d'une catégorie d'un paramètre crucial et un paramètre non crucial

3.3 Étapes de la méthodologie CKIP

Comme mentionné précédemment, la théorie des catégories est utilisée pour fournir un cadre formel permettant d'identifier les connaissances les plus importantes qui doivent être capitalisées et partagées au cours du processus de collaboration. Notre méthodologie que nous appelons *Crucial Knowledge Identification Process* (CKIP) est composée de cinq étapes principales [116]. Les différentes étapes de l'approche proposée sont décrites dans la figure (3.3).

3.3.1 Étape 1 : Création des catégories pour les paramètres identifiés dans les exigences

Dans un premier temps, le manager de projet crée des catégories pour chaque paramètre spécifié dans les exigences. Ces catégories sont considérées comme le point de départ du processus d'identification des connaissances cruciales. A la fin de cette étape, nous obtenons un ensemble de catégories PC (catégorie P1, catégorie P2 par

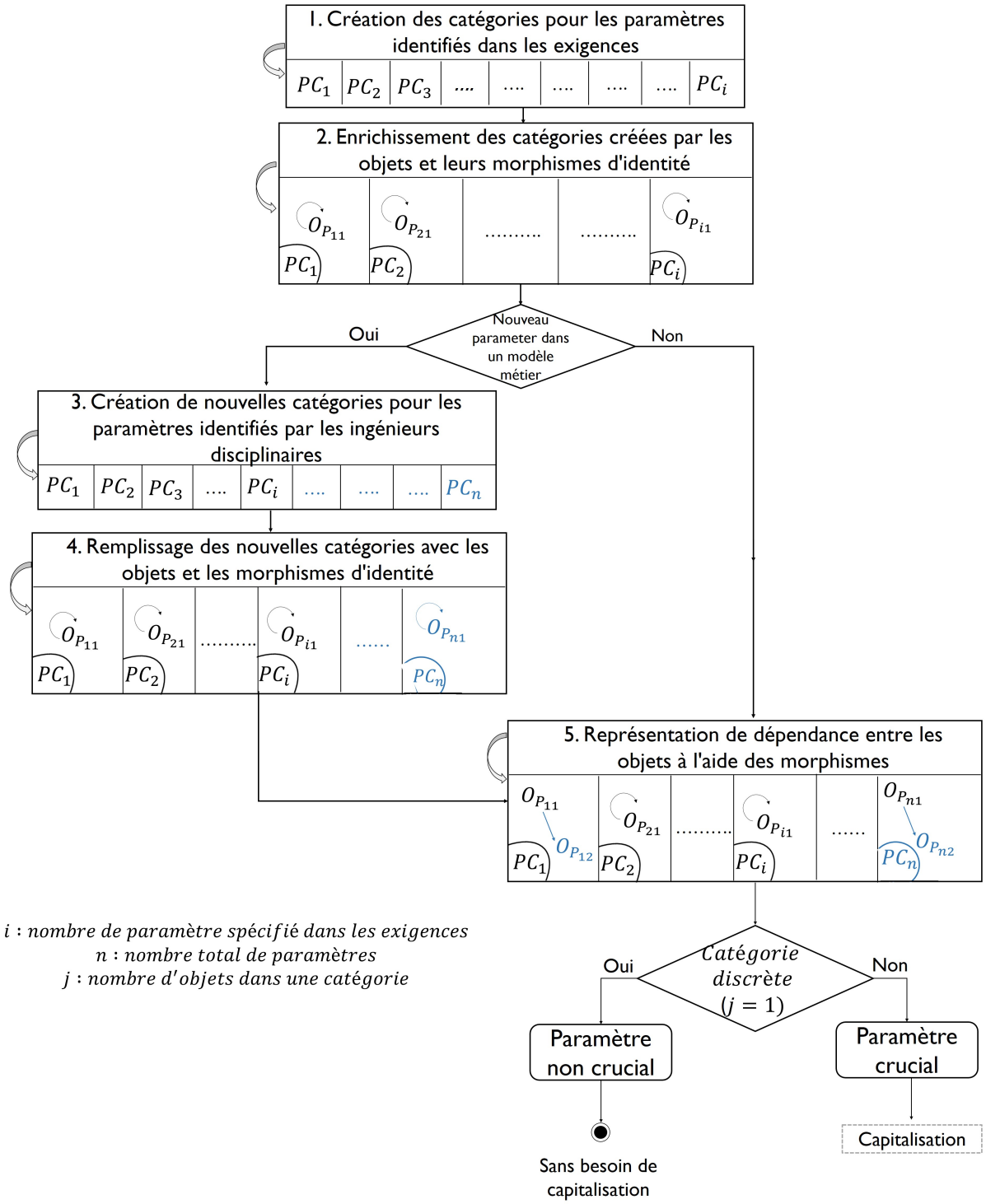


FIGURE 3.3 – Les étapes de la méthodologie CKIP

exemple dans la figure (3.2) où i est le nombre de paramètres identifiés dans les exigences.

3.3.2 Étape 2 : Enrichissement des catégories créées par les objets et leurs morphismes d'identité

A ce niveau, le manager de projet doit remplir les catégories créées avec les objets nécessaires. Les objets OPi_1 représentent la première instance du paramètre qui fait référence au modèle métier initial qui utilisera le paramètre concerné. De plus, pour chaque objet d'une catégorie, il existe un morphisme d'identité (la flèche en boucle où la source et la cible sont le même objet). Ce morphisme est une représentation naturelle de l'évolution interne du paramètre.

3.3.3 Étape 3 : Création de nouvelles catégories pour les paramètres identifiés par les ingénieurs disciplinaires

Ici, si les ingénieurs disciplinaires ont besoin d'un autre paramètre qui n'est pas défini dans les exigences, une nouvelle catégorie représentant le paramètre nécessaire est créée. Tous les paramètres impliqués dans le processus de collaboration doivent être représentés sous forme de catégories.

3.3.4 Étape 4 : Remplissage de nouvelles catégories avec les objets et les morphismes d'identité

Comme à la deuxième étape, à ce niveau, les catégories créées seront remplies avec les objets correspondants et leurs flèches d'identité. Les étapes (3) et (4) seront répétées jusqu'à obtenir des catégories pour chaque paramètre utilisé par les modèles métiers (PC_n, où n représente le nombre total de paramètres).

3.3.5 Étape 5 : Représentation de dépendance entre les objets à l'aide des morphismes

Dans cette dernière étape, tous les ingénieurs mettent en évidence l'instanciation des paramètres nécessaires en créant un nouvel objet dans la catégorie correspondante et en reliant cet objet au précédent par un morphisme. Si la catégorie contient plus qu'un objet ($j > 1$, le paramètre évolue d'un modèle métier à un autre), le paramètre est considéré comme crucial et nécessite la capitalisation. Dans le cas contraire, il ne sera pas capitalisé et reste spécifique à un seul modèle métier.

3.4 Comparaison entre la méthodologie CKIP et les méthodologies existantes

Comme nous avons indiqué précédemment, La capitalisation des connaissances est une opération importante pour le succès de l'entreprise. Afin d'optimiser cette opération, il est nécessaire de se concentrer uniquement sur les connaissances cruciales. En ce sens, divers travaux de recherche ont été proposés afin d'étudier l'identification des connaissances cruciales. Dans le travail de recherche présenté par Pomian et Roche [62], l'extraction des connaissances cruciales est réalisée par le biais de réunions entre les différentes équipes de conception impliquées dans la collaboration.

Le même concept est utilisé par Mcharek et al. [7]. Dans leur travail de recherche, les auteurs ont fait valoir qu'une réunion entre les différentes équipes de conception peut être organisée pour extraire les connaissances cruciales. Cette solution informelle nécessite un temps considérable pour être réalisée.

La principale différence entre ces solutions et l'approche que nous proposons est que, dans cette dernière, le processus d'identification repose sur la base formelle de la théorie des catégories. D'une part, ce cadre mathématique formel unifie les modèles hétérogènes impliqués dans le processus de conception, ce qui convient à la conception collaborative mécatronique où un environnement de conception homogène est nécessaire. D'autre part, cette formalisation des connaissances évite la nécessité d'un échange direct entre les parties prenantes, qui peut être difficile à organiser dans un environnement complexe tel que celui de la conception mécatronique où un grand nombre de connaissances est utilisé.

L'utilisation de la TC comme outil de formalisation fournit non seulement une base formelle de connaissances mais aussi un cadre compréhensible pour toutes les équipes de conception, en utilisant des catégories et des morphismes pour représenter les connaissances. En outre, les avantages de la formalisation des connaissances par la TC peuvent être prouvées à long terme. La sauvegarde des connaissances cruciales sous des formes structurées et organisées (c'est-à-dire les catégories dans notre approche CKIP) facilitera la traçabilité de la collaboration, ce qui est important dans la conception des systèmes mécatroniques. Assurer cette traçabilité permettra d'identifier qu'une connaissance donnée a été utilisée ou modifiée par un ingénieur donné à un moment donné, ce qui facilitera la détection des conflits. De plus, la traçabilité de l'évolution de la collaboration facilite les phases de réutilisation dans des projets de conception futurs.

3.5 Validation de la méthodologie CKIP sur un boîtier papillon

3.5.1 Présentation du boîtier papillon

Le boîtier papillon ou en anglais *Electronic Throttle Body* (ETB) est un système mécatronique permettant le réglage du débit d'air entrant dans la chambre à combustion interne d'un moteur d'automobile. Le modèle commun de l'ETB, illustré par la figure (3.4), est constitué généralement d'un moteur DC, un réducteur, un papillon des gaz (un clapet), deux ressorts pour la position «*limp-home*» et un capteur de position.

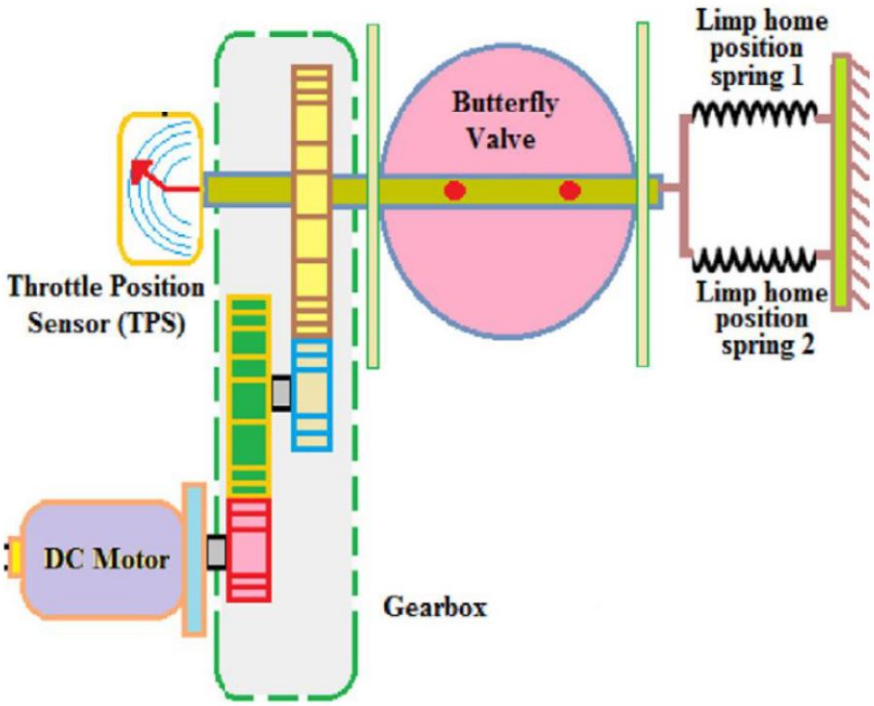


FIGURE 3.4 – Configuration d'un boîtier papillon [117]

Afin de réduire l'erreur de position du clapet, un système de contrôle à rétroaction en boucle fermée est réalisé en employant un capteur de position du papillon [117]. Ce capteur mesure la position du papillon par deux méthodes différentes avec et sans contact. Le type avec contact utilise un potentiomètre à double redondance pour mesurer la position. La corrélation et le suivi des deux capteurs peuvent être utilisés pour détecter les défaillances dans le système, les connexions électriques, etc. Les capteurs de position utilisés dans la plupart des véhicules sont constitués d'un

potentiomètre qui fournit un signal de sortie proportionnel à la position de l'arbre du papillon. Cependant, le capteur de position sans contact peut fonctionner avec trois principes différents : L'effet Hall, magnéto-résistif et inductif. En se basant sur ces principes, le mouvement rotatif du papillon est converti en une tension de sortie linéaire directement proportionnelle à l'angle de rotation.

L'ETB comporte deux ressorts intégrés pour maintenir le clapet ouvert à un angle par défaut, cette position est appelée position de repos ou *"limp-home position"*. Le premier ressort permet d'amener le clapet de la position ouverte à la position *limp-home* et le deuxième assure le passage de la position fermée à la position *limp-home*. Cela permet d'assurer toujours la faible quantité d'air nécessaire pour faire fonctionner le moteur dans des conditions fixes même en cas de défaillance dans l'ETB. Grâce à cette solution, la position de fermeture complète du clapet est évitée, ce qui permet au conducteur de « boiter » jusqu'à ce qu'il arrive à la station-service la plus proche en cas de panne [118].

3.5.2 Développement pluridisciplinaire de l'ETB

Afin de valider notre approche CKIP, nous avons créé quatre modèles métiers : un modèle d'exigences (EM_R), un modèle multi-physique (EM_{MP}), un modèle de choix des composants (EM_C) et un modèle 3D (EM_{3D}).

Modèle d'exigences

Un modèle d'exigences ou *Requirement model* (EM_R) permet d'identifier les exigences qui doivent être respectées durant le processus de conception de l'ETB. Le modèle d'exigences, illustré par la figure (3.5), est créé à l'aide du langage SysML dans l'environnement Magic Draw.

Modèle multi-physique

Un modèle multi-physique ou *Multi-physical model* (EM_{MP}) est créé pour effectuer une analyse dynamique avec le langage Modelica en utilisant l'environnement Dymola comme le montre la figure (3.6). Ce modèle contient un moteur à courant continu connecté à un contrôleur PID (Proportionnelle - Intégrale - Dérivée), un réducteur qui amplifie le couple délivré par le moteur, un clapet, un modèle de friction, un ressort (*Stop spring*) limitant la rotation du clapet entre 0° et 90°, un couple aérodynamique représentant la différence de pression entre la pression ambiante et la pression de combustion et deux ressorts *Limp Home*. Le ressort principal ($LHspring_1$) amène le clapet de la position ouverte à la position *Limp Home* et le second amène le clapet de la position fermée à la position *Limp Home*.

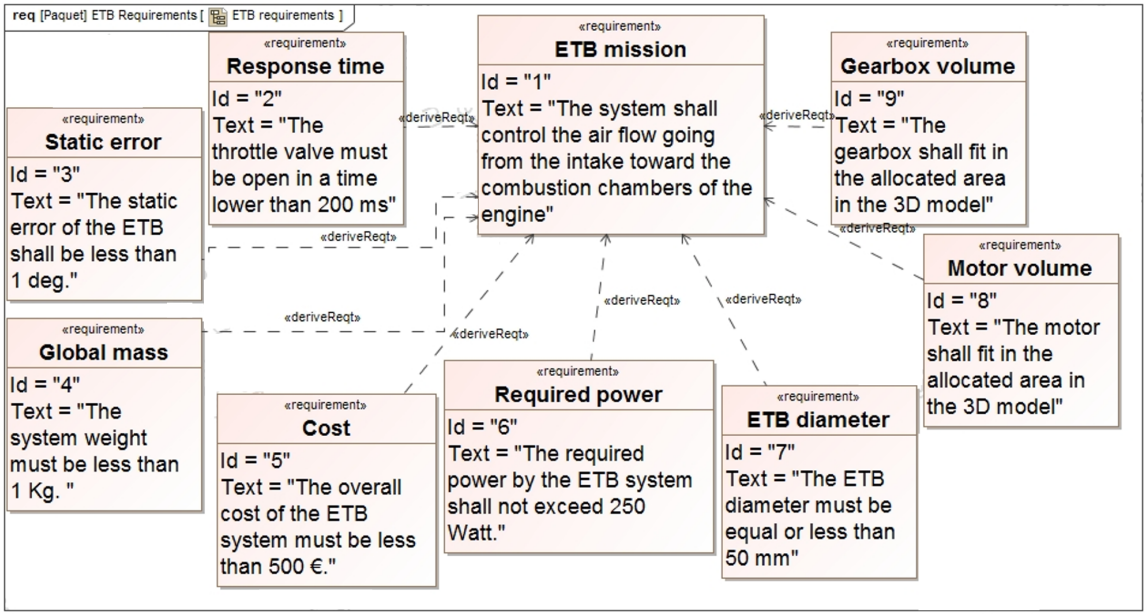


FIGURE 3.5 – Modèle d'exigences du boîtier papillon

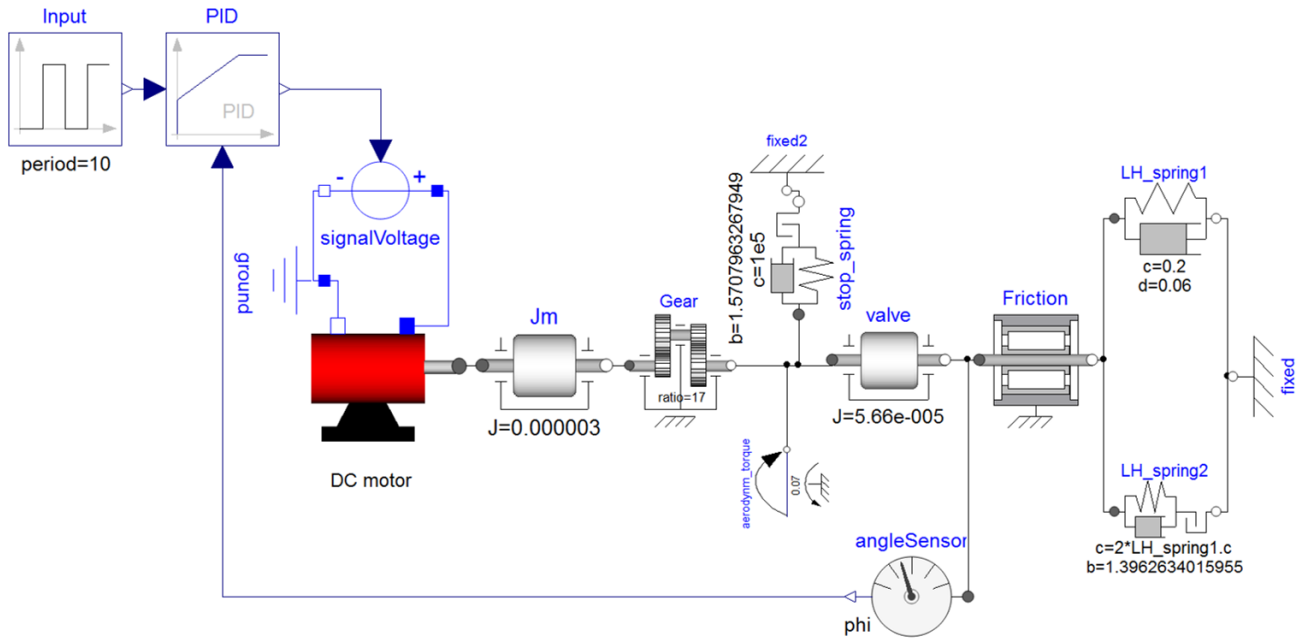


FIGURE 3.6 – Modèle multi-physique du boîtier papillon

Modèle de choix des composants

Un modèle de choix des composants ou *Commercial Off-The-Shelf model COTS* (EM_C) qui contient les caractéristiques des composants de l'ETB. Ces composants seront choisis en fonction des résultats de la simulation effectuée dans le modèle multi-physique. Un extrait des caractéristiques du moteur DC est représenté par le tableau (3.1).

TABLE 3.1 – Modèle de choix des composants (EM_C) du boîtier papillon (extrait des caractéristiques du moteur DC [119])

Valeurs à la tension nominale	
Tension nominale	12 V
Vitesse à vide	8130 tr/min
Courant à vide	320 mA
Vitesse nominale	7610 tr/min
Couple nominal	77,7 mN.m
Rendement maximal	86%

Modèle 3D

un modèle 3D (EM_{3D}) en utilisant le logiciel CATIA qui permet de vérifier l'intégration des composants sélectionnés dans la zone allouée (voir figure 3.7).

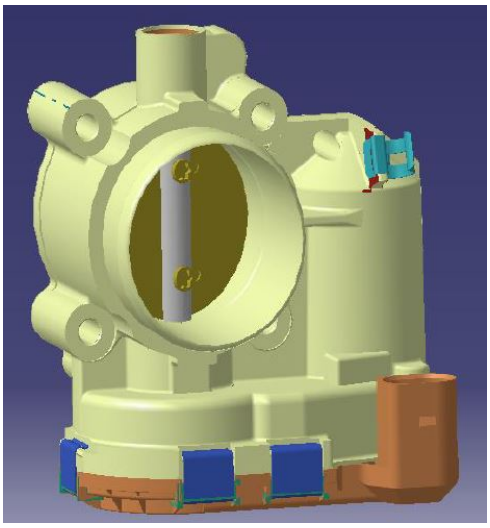


FIGURE 3.7 – Modèle 3D du boîtier papillon

Même si on s'intéresse à la conception collaborative, il faut tenir compte d'un *workflow* bien défini lors de la création de chaque modèle métier. Ce flux commence par le modèle d'exigences (EM_R) suivi du modèle multi-physique (EM_{MP}). Une fois les résultats de la simulation obtenus à partir du modèle multi-physique, les composants COTS (EM_C) seront trouvés. Le dernier modèle (EM_{3D}) est utilisé pour vérifier l'intégration 3D du mécanisme.

À ce niveau, les modèles métiers de l'ETB sont établis. Passons, ainsi, à l'application de notre approche sur l'ETB.

3.5.3 Étape 1 : Création des catégories pour les paramètres identifiés dans les exigences durant la conception de l'ETB

La première étape de notre méthodologie consiste à traduire les exigences initiales identifiées par le chef de projet en catégories. L'idée est de considérer les paramètres comme des catégories et leurs différentes instances comme des objets. L'instanciation d'un paramètre d'un modèle métier à un autre est établie par des morphismes. Les catégories créées pour les paramètres identifiés dans les exigences sont représentées par la figure (3.8).

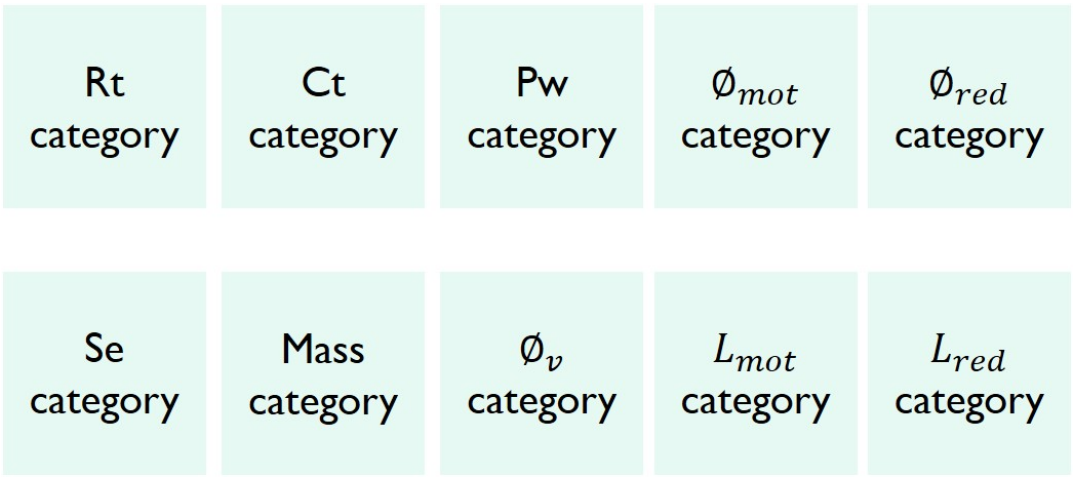


FIGURE 3.8 – Catégories pour les paramètres identifiés dans les exigences avec Rt : temps de réponse, Ct : le coût, Pw : la puissance maximale, Ømot, Øred : diamètres du moteur et réducteur respectivement, Se : l'erreur statique, $Mass$: la masse globale, Øv : diamètre de la valve, L_{mot} et L_{red} les longueurs du moteur et réducteur respectivement

3.5.4 Étape 2 : Enrichissement des catégories créées par les objets et leurs morphismes d'identité

Une fois que toutes les catégories requises ont été créées, les objets initiaux ainsi que leurs morphismes d'identité doivent être représentés dans les catégories concernées. Ces objets initiaux illustrent la première instance du paramètre proposée par le modèle d'exigences (EM_R). Comme mentionné précédemment, le morphisme d'identité de chaque objet met en évidence l'évolution interne de l'état du paramètre dans le même modèle.

3.5.5 Étape 3 : Création de nouvelles catégories pour les paramètres identifiés par les ingénieurs disciplinaires

Après avoir créé les catégories des paramètres identifiés dans les exigences, d'autres catégories sont également créées (voir figure 3.9). Des catégories pour l'inertie (J_{mot}), la résistance (R_{mot}) et l'inductance du moteur (L_{mot}) ainsi que le ratio du réducteur (r_{red}) sont établies.

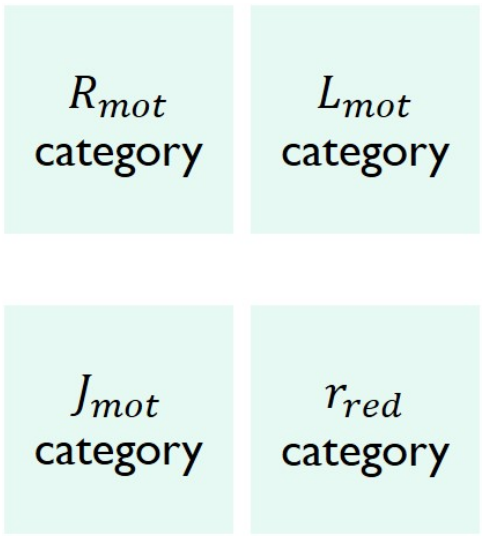


FIGURE 3.9 – Catégories de l'inertie, la résistance, l'inductance du moteur et le ratio du réducteur

3.5.6 Étape 4 : Remplissage des nouvelles catégories avec les objets et les morphismes d'identité

Chaque catégorie sera remplie avec les objets et leurs flèches d'identité. Prenons l'exemple de l'erreur statique (voir figure 3.10.a) définie dans le modèle d'exigences (EM_R) et la puissance du moteur représentée par la catégorie *Pw Category* comme le montre la figure (3.10.b).

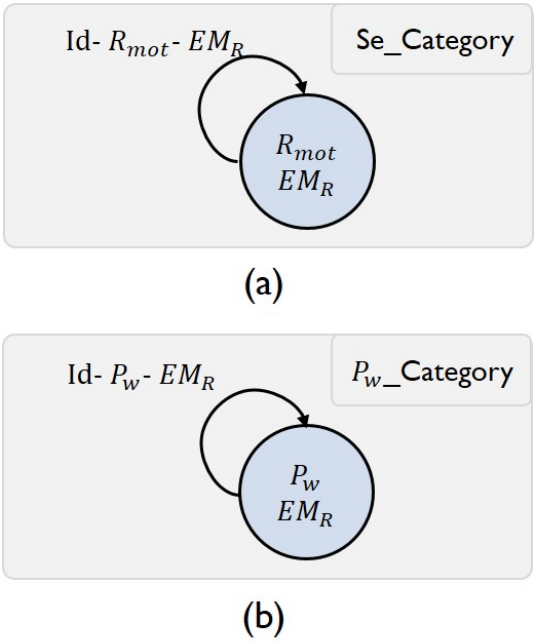


FIGURE 3.10 – Catégories de l'erreur statique et la puissance du moteur

3.5.7 Étape 5 : Représentation de dépendance entre les objets à l'aide des morphismes

Finalement, chaque catégorie créée au début sera modifiée par les différents ingénieurs. Cette modification consiste à ajouter un nouvel objet qui représente une autre instance du même paramètre et à relier cet objet à celui précédemment créé par un morphisme.

Tenons l'exemple de la catégorie de la puissance du moteur (P_w). Dans cette catégorie trois objets sont mis en valeur. Le premier objet correspond à la première instanciation du paramètre P_w dans le modèle d'exigences alors que le deuxième représente la deuxième instanciation dans le modèle multi-physique. Le dernier objet est une autre instanciation dans le modèle COTS. La puissance du moteur est représentée par la catégorie *Pw Category* comme nous pouvons le constater sur la figure

(3.11.a). Par ailleurs, la catégorie de la résistance du moteur contiendra deux objets, le premier étant créé par l'ingénieur travaillant sur le modèle d'exigences (EM_R) et le second étant élaboré par la simulation dans le modèle multi-physique (EM_{MP}) (voir 3.11.b). La catégorie de l'inductance du moteur, illustrée par la figure (3.11.c), contient également deux objets représentant deux instances différentes de l'inductance. Cependant, les catégories du rendement du moteur ainsi que l'axe de rotation sont représentées par les figures (3.11.d) et (3.11.e). Ces catégories contiennent uniquement un seul objet. Cela signifie que le paramètre correspondant est instancié par un seul ingénieur disciplinaire et dans un seul modèle métier.

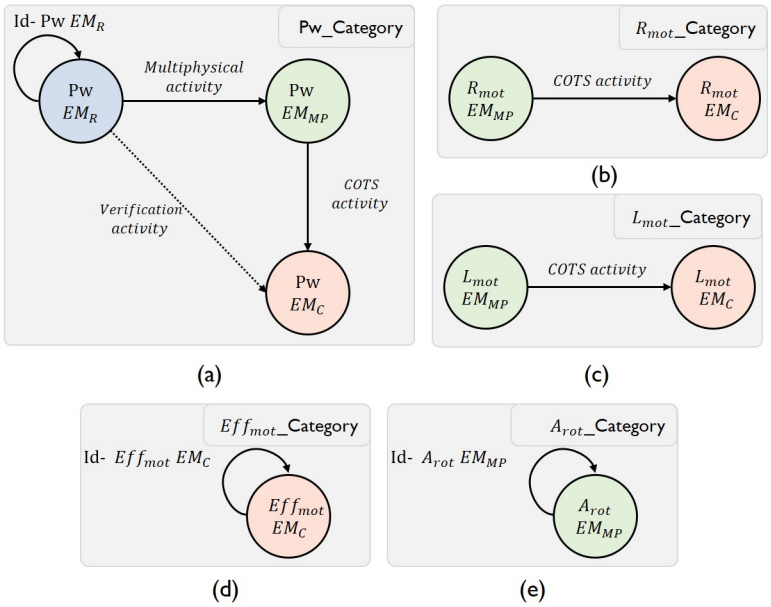


FIGURE 3.11 – Les différentes catégories avec les différents objets

Tout ce dont nous avons besoin pour garantir que la catégorie créée respecte la définition mathématique d'une catégorie est de prouver l'existence de la composition, de l'identité ainsi que de l'associativité. Pour cela nous allons tenir comme exemple la catégorie de la puissance du moteur représentée par la figure (3.11.a). Soit Pw_{EMR}, Pw_{EMMP} et Pw_{EMC} trois instances différentes de la puissance maximale telles que EM_R interagit avec EM_{MP}, qui à son tour interagit avec EM_C. Ensuite, EM_R peut interagir avec EM_C indirectement à travers EM_{MP}, ce qui met en évidence l'existence d'une composition de morphismes entre EM_R et EM_C. Le morphisme d'identité existe pour représenter l'évolution interne du paramètre Pw. On suppose qu'elle est présente pour chaque objet. Cependant, elle est rarement montrée afin d'éviter d'encombrer la catégorie avec un morphisme d'identité pour chaque objet. Soit *multiphysical activity*, *COTS activity* et *verification activity* trois morphismes tels que *multiphysical activity* : $Pw_{EMR} \rightarrow Pw_{EMMP}$, *COTS activity* : $Pw_{EMMP} \rightarrow$

Pw_{EMC} et *verification activity* : $Pw_{EMR} \rightarrow Pw_{EMC}$. Il est clair que *multiphysical activity* $\circ$ (*COTS activity* $\circ$ *verification activity*) = (*multiphysical activity* $\circ$ *COTS activity*) $\circ$ *verification activity*. Ces propriétés sont prouvées pour les catégories illustrées par les figures (3.11.b) et (3.11.c). Cependant, les catégories dans les figures (3.11.d) et (3.11.e) sont des catégories discrètes (contenant uniquement le morphisme d'identité), où les propriétés mathématiques d'une catégorie ne peuvent pas être prouvées.

Après avoir créé toutes les catégories nécessaires, le chef de projet doit identifier les paramètres cruciaux. Comme mentionné précédemment, les paramètres dans lesquels les catégories correspondantes ont la forme d'une catégorie discrète seront considérés comme non cruciaux pour le processus de collaboration et n'ont pas besoin d'être capitalisés. L'idée est de considérer la contribution de ces paramètres comme non cruciale pour atteindre les objectifs du projet puisqu'ils sont utilisés ou créés par un modèle métier unique. Ainsi, les autres modèles ne les utiliseront pas dans leurs activités d'ingénierie. Cependant, les catégories restantes, qui contiennent au moins deux objets (c'est-à-dire au moins deux instances du même paramètre), seront prises en considération et les paramètres correspondants seront définis comme cruciaux pour la collaboration. Tous les paramètres générés par les quatre modèles métiers impliqués dans la conception de l'ETB sont regroupés dans le tableau (3.2). Ce tableau présente la classification finale des paramètres. Les connaissances cruciales et celles qui ne le sont pas sont identifiées d'une manière simple et formelle. La classe $KClass_1$ contiendra les paramètres qui devraient faire l'objet d'une capitalisation dans le processus collaboratif. Cependant, la classe $KClass_2$ regroupera les paramètres non cruciaux.

TABLE 3.2 – Classification finale des paramètres utilisés dans la conception de l'ETB

Paramètre crucial ($KClass_1$)	Paramètre non crucial ($KClass_2$)
Rt, Se, Mass, Pw, Ct, Ømot Lgth_{mot}, $Rmot$, $Lmot$, $Jmot$,Ørred Lgth_{red},r_{red}	Eff_{mot},Eff_{red}, A_{rot}

La théorie des catégories a été utilisée dans notre méthodologie nommée CKIP comme un outil d'unification et de formalisation. A travers cette théorie, les modèles métiers hétérogènes ont été unifiés pour faciliter l'identification des connaissances cruciales. Dans notre approche, chaque paramètre est représenté par une catégorie où ses différents objets sont les instances du paramètre correspondant. Cette représentation nous a fourni un cadre formel pour extraire les connaissances cruciales. En effet, les paramètres dont les catégories contiennent au moins deux objets seront considérés cruciaux. Cependant, les autres paramètres qui sont représentés par des catégories discrètes seront considérés non cruciaux à la collaboration et ne nécessiteront pas la capitalisation.

Dans la section suivante, nous allons présenter notre méthodologie que nous désignons par *Conflict Resolution Process* (CRP) pour détecter et gérer les conflits durant la conception collaborative.

3.6 La méthodologie Conflict Resolution Process (CRP)

Le processus de conception collaborative fait intervenir des parties prenantes issues de différents milieux et domaines, dans le but d'atteindre un objectif commun concernant le développement du produit [58]. Ces experts ont des points de vue différents sur le système à concevoir et utilisent des langages et des outils de modélisation hétérogènes. Cependant, les interrelations entre ces modèles sont inévitables. Par conséquent, des conflits peuvent apparaître et doivent être soigneusement gérés. Pour cela, nous proposons une nouvelle méthodologie nommée *Conflict Resolution Process* (CRP), basée sur la théorie des catégories, visant à diagnostiquer et à résoudre les incohérences entre les modèles métiers impliqués dans le processus de collaboration. D'une part, notre approche est basée sur des règles et des *patterns*, ce qui la rend plus flexible. Cette technique permet de modifier, supprimer ou ajouter des nouvelles règles pendant le processus de résolution des conflits, ce qui peut réduire le temps de développement dans la conception mécatronique. D'autre part, nous utilisons la théorie des catégories dans notre méthodologie comme un cadre formel pour illustrer la détection et la résolution des conflits. La motivation de l'utilisation de cette théorie est la puissance de ses constructions mathématiques. En utilisant cette théorie, les graphes porteront toutes les intuitions issues de la pratique et auront une signification formelle. De plus, la traçabilité de la collaboration entre les différents acteurs est assurée de manière formelle par le biais de différentes catégories contenant tous les résultats de cet échange. De plus, la localisation des conflits dans le processus de résolution des conflits est basée sur la mise en correspondance ou le mapping entre les catégories en utilisant des foncteurs, ce qui rend cette action formelle et facile à la fois.

3.7 Architecture de la méthodologie CRP

L'architecture de notre approche est illustrée par la figure (3.12). Dans un premier temps, les différents paramètres des modèles métiers doivent être représentés sous forme d'instances de catégories des paramètres ou *Parameter Category Instance* (PCI). Cette étape fournit un formalisme commun qui rend les différents modèles métiers homogènes malgré leur hétérogénéité. Ensuite, les dépendances entre les paramètres sont établies au moyen de la TC par la définition des morphismes représentant le coefficient de dépendance entre les différentes instances d'un paramètre (c'est

à dire les différents objets). À ce niveau, le processus de détection et de résolution peut commencer. Un ensemble de règles de cohérence ou *Consistency Rule* (CR) est établi pour identifier les contraintes à respecter. En ce qui suit, les conflits apparus au cours de la collaboration entre les modèles métiers sont détectés par le mapping entre les catégories des *patterns* et les catégories des paramètres. Si le paramètre en conflit a des liens de dépendances avec un autre, le paramètre correspondant doit être mis à jour. Par conséquent, les catégories des paramètres correspondantes doivent également être mises à jour et les valeurs modifiées seront enregistrées dans les catégories des paramètres mises à jour ou *Updated Parameter Category Instance* ($UPCI$). Sinon, l'une des actions de résolution ou *Resolution Action* (RA) est appliquée au paramètre en conflit afin de gérer cette contradiction. Les résultats finaux du processus de résolution des conflits seront sauvegardés dans les instances finales des catégories des paramètres ou *Final Parameter Category Instance* ($FPCI$) pour assurer la traçabilité de la collaboration.

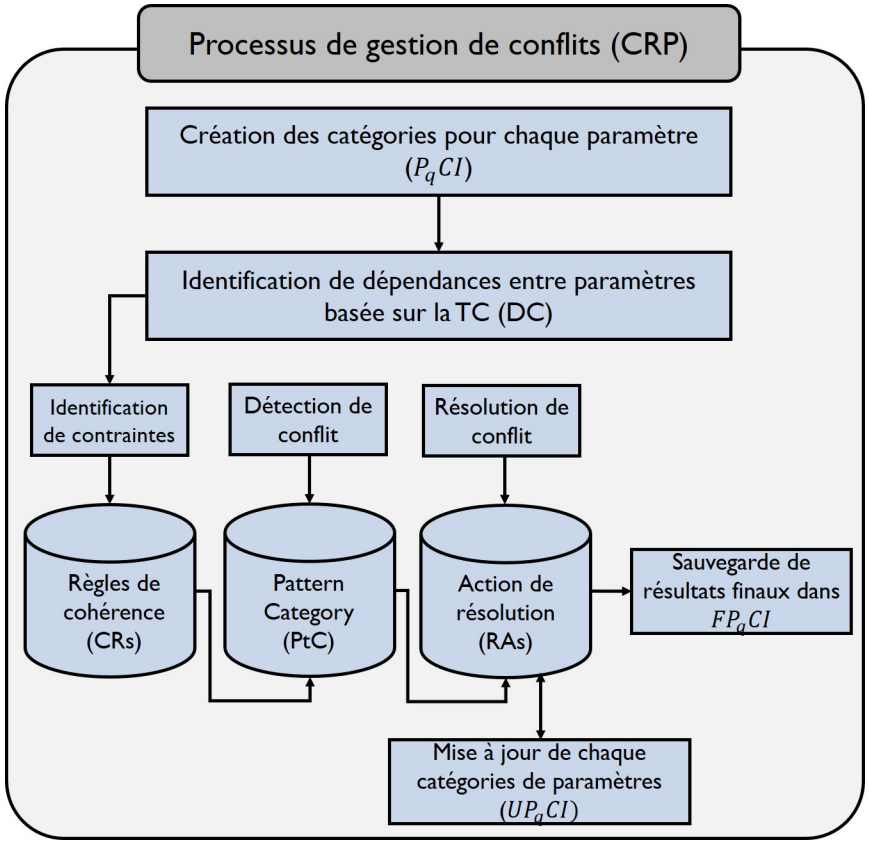


FIGURE 3.12 – Architecture de la méthodologie CRP

3.7.1 Détail du concept Parameter Category Instance (PCI)

Parameter Category Instance (PCI) représente les catégories des paramètres, P_1CI, P_2CI, ...,P_mCI, qui contiennent des paramètres cruciaux extraits des modèles métiers et aideront à capturer les conflits entre ces modèles. La catéogrie PCI est représentée sous forme d'un ensemble de cinq éléments PCI= $< O, id, A_r, L_o, L_{Ar} >$ est une catégorie, où O = (O_1,..., O_j) est un ensemble d'objets. Chaque objet a son propre morphisme d'identité (c'est-à-dire la flèche bouclée). Ces objets sont les différentes instances du paramètre. Les objets sont reliés les uns aux autres par un ensemble de flèches ou de morphismes A_r (*Arrow*). Tous les objets et flèches ont des attributs (L_o et L_{Ar} respectivement). Un exemple de P_1CI créée selon la définition précédente de la catégorie, est illustré par la figure (3.13). Les morphismes d'identité ainsi que les flèches de composition ne sont pas représentés dans cette catégorie afin d'éviter de l'encombrer avec un grand nombre de morphismes.

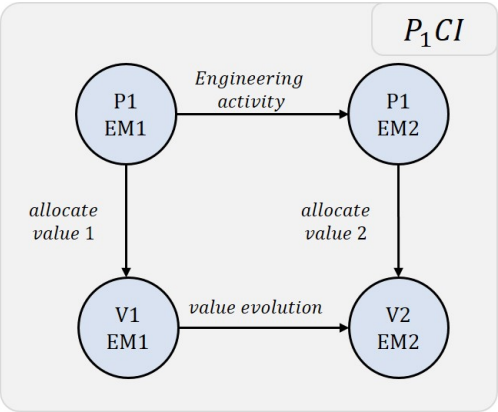


FIGURE 3.13 – Parameter Category Instance (PCI). P_{1EM1}, P_{1EM2} ,V_{1EM1}, V_{2EM2} sont les objets du P_1CI. $allocatevalue_1$ $allocatevalue_2$, *valueevolution* sont les morphismes de P_1CI. Les morphismes d'identité et de composition ne sont pas représentés dans cette catégorie

3.7.2 Détail du concept Updated PCI

Cette catégorie contiendra les valeurs actualisées obtenues après chaque itération du processus de résolution des conflits. Comme le P_mCI décrit précédemment, le UP_mCI est considéré comme une catégorie et hérite de toutes les propriétés de P_mCI. Grâce à ces catégories mises à jour, la traçabilité peut être assurée, ce qui sera utile dans des perspectives de réutilisation.

3.7.3 Détail du concept Final PCI

Cette catégorie contient les résultats finaux obtenus après avoir appliqué les actions de résolution appropriées aux conflits détectés. De même, cette catégorie a la même forme que les P_mCI et UP_mCI décrites précédemment.

3.7.4 Détail du concept Dependency Category (DC)

La catégorie de dépendance ou *Dependency Category* (DC) illustre les dépendances existantes entre les paramètres présentés dans les différentes catégories des paramètres en se basant sur la théorie des catégories. Un ensemble de 5 éléments DC = $<O_d$, id_d, Ar_d, Lo_d, $L_{Ard}>$ est une catégorie de dépendance où O = (O_1,..,O_m) est un ensemble d'objets où m se réfère au nombre de paramètres utilisés dans le processus de conception. Chaque objet a son propre morphisme d'identité. Le morphisme d'identité représente l'évolution interne de chaque objet. On suppose qu'il est présent pour la DC. Cependant, il est rarement montré afin d'éviter d'encombrer la catégorie avec un morphisme d'identité pour chaque objet. Dans cette catégorie, les objets sont les différents paramètres utilisés par les experts impliqués dans la conception. Les objets sont reliés entre eux par un ensemble de flèches ou morphismes Ar_d. Ces morphismes mettent en évidence la dépendance entre les différents paramètres. Tous les objets et flèches ont des attributs Lo_d et LAr_d respectivement. Les attributs des morphismes représentent le coefficient de dépendance de chaque relation entre deux objets. Ces coefficients seront utilisés dans le processus de résolution des conflits. Un exemple d'une catégorie de dépendance est illustré par la figure (3.14).

3.7.5 Détail du concept Consistency Rule (CR)

Règle de cohérence ou *Consistency Rule* (CR) un ensemble de règles de cohérences, CR_1, CR_2, ... , CR_p, entre les éléments dans les modèles métiers. Ces modèles seront considérés comme incohérents si les CRs ne sont pas respectées.

3.7.6 Détail du concept Pattern Category (PtC)

Les catégories de *pattern* ou *Pattern Category* (PtC), PtC_1, PtC_2,...,PtC_q, représentent un ensemble de catégories visant à décrire l'existence d'un conflit. Le mapping entre les catégories des paramètres et la catégorie d'un pattern permet de détecter les conflits. Ce pattern est défini comme une catégorie contenant un ensemble d'objets et de morphismes. Les objets de cette catégorie sont soit une chaîne de caractères représentant les noms des paramètres, soit une variable pour la définition des valeurs des paramètres. Les morphismes dans la PtC illustrent l'attribution

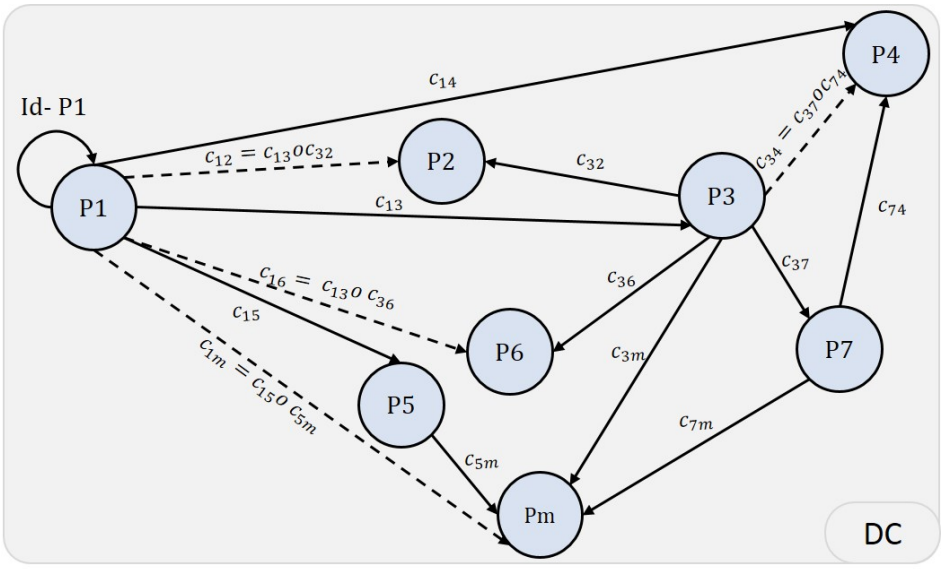


FIGURE 3.14 – Catégorie de dépendance (DC). P_1, P_2, ...,P_m sont les objets de DC. c_{12}, c_{32}, c_{13}, etc. représentent les morphismes. Le morphisme d'identité n'est représenté que pour P_1 mais il est supposé être présent pour tous les paramètres. Les flèches en pointillés font référence à la composition des morphismes

de valeurs pour chaque paramètre. La forme du PtC peut varier en fonction de la catégorie des paramètres à interroger. Un exemple de PtC est illustré à la figure (3.15). La PtC_1 décrit l'existence d'un conflit entre deux modèles métiers où la valeur de m_1 n'est pas égale ou inférieure à m_2. Le second PtC_2 représente un conflit entre trois modèles métiers où m_1 n'est pas égal ou inférieur à m_2 et m_3. De plus, un conflit peut être détecté lorsque les valeurs de m_2 et m_3 ne sont pas égales.

3.7.7 Détail du concept Resolution Action (RA)

Ensemble d'actions, RA_1, RA_2, ... , RA_r, destinées à résoudre les conflits détectés. Ces actions couvrent la modification de la valeur du paramètre en conflit, la tolérance du conflit et son ignorance. Le choix de ces actions est assuré par le chef de projet et dépend du contexte actuel ainsi que des conflits détectés.

Dans la section suivante, les différentes étapes de notre méthodologie CRP seront détaillées.

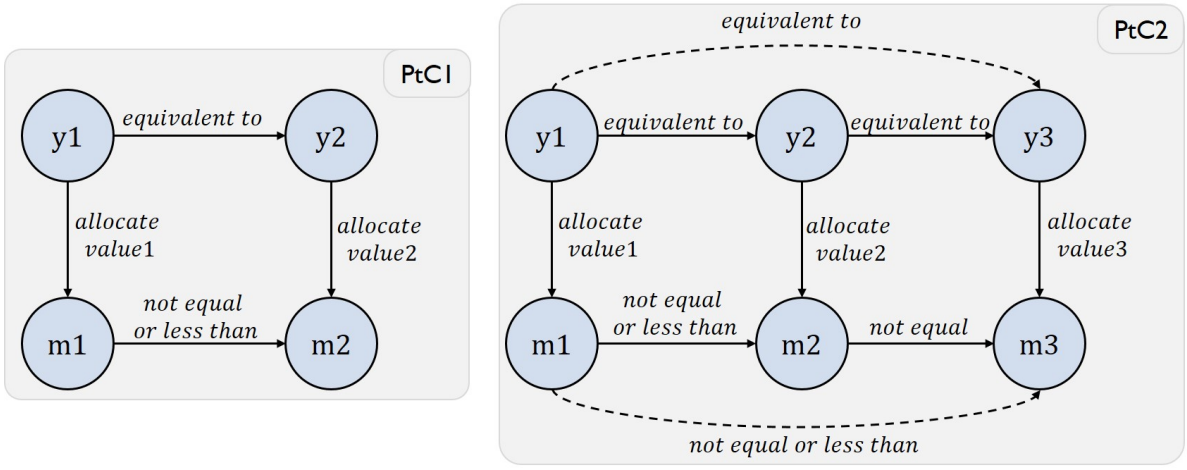


FIGURE 3.15 – Catégorie de pattern (PtC). y_1, y_2 et y_3 sont des objets constants représentant les noms des paramètres. m_1 m_2 et m_3 sont des objets variables pour définir les valeurs des paramètres

3.8 Étapes de la méthodologie CRP

Notre méthodologie est composée de sept étapes principales [120]. L'organigramme de l'approche proposée est présenté dans la figure (3.16).

3.8.1 Étape 1 : Création d'un formalisme commun basé sur la théorie des catégories pour la représentation des paramètres *Parameter Category Instance* (PCI)

Selon l'architecture dirigée par les modèles ou *Model-Driven Architecture* (MDA) introduite par l'*Object Management Group* (OMG) [121], un modèle peut être défini comme une représentation de la réalité et une abstraction de choses significatives et pertinentes pour les parties prenantes, décrites à l'aide des langages bien définis [112]. Chaque modèle contient une variété de connaissances relatives à des modèles métiers spécifiques. Ces connaissances définissent la manière dont les différents éléments sont liés les uns aux autres. Ainsi, un modèle peut être considéré comme un ensemble d'éléments et de relations. Pour cette raison, il est naturel de considérer les modèles comme des graphes. Ainsi, au lieu de travailler directement sur les langages des modèles hétérogènes, un formalisme unifiant est requis. Ces graphes ont une signification formelle dans le contexte de la TC et mettent en évidence toutes les intuitions issues de la pratique. La première étape de notre méthodologie consiste à établir un cadre formel et complet au moyen de cette théorie mathématique. Herzig [122] affirme que l'unification de tous les paramètres contenus dans les modèles

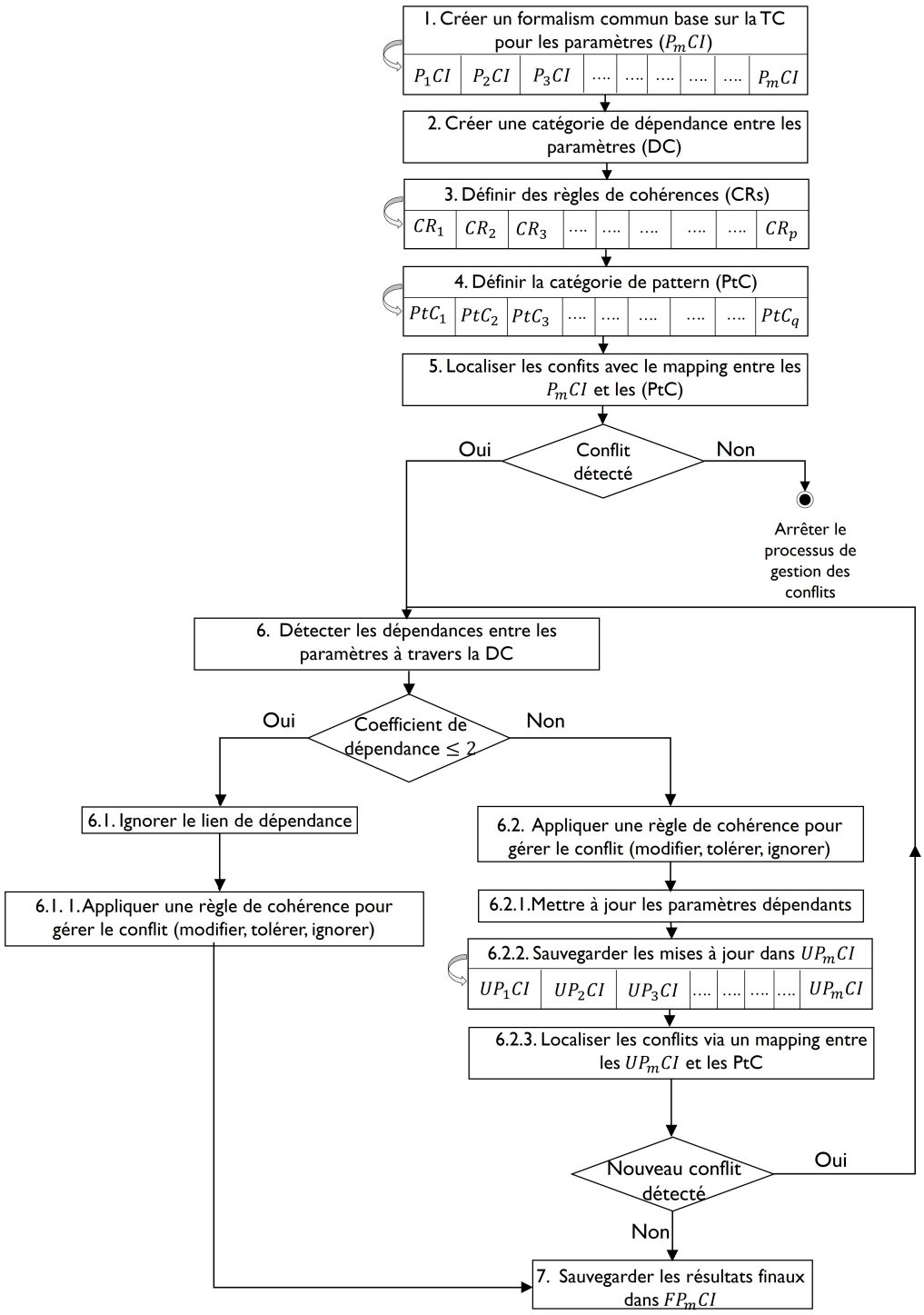


FIGURE 3.16 – Organigramme de l'approche de résolution des conflits basée sur la théorie des catégories

n'est pas utile. Par conséquent, seuls les paramètres cruciaux seront pris en compte et seront également traités dans la résolution des conflits. Ainsi, pour chaque paramètre crucial, une catégorie de paramètre (P_mCI) est créée. L'idée est de considérer les paramètres comme des catégories, les objets comme les différentes instances du paramètre et les morphismes comme la progression du paramètre concerné d'un modèle métier à un autre. A la fin de cette étape, on obtient un ensemble de catégories P_mCI où m représente le nombre de paramètres cruciaux impliqués dans la conception collaborative [123].

3.8.2 Étape 2 : Création d'une catégorie de dépendance entre les paramètres

Au cours de la conception collaborative, des interrelations entre les modèles métiers ainsi que leurs éléments sont susceptibles de se produire. Ces interrelations peuvent causer plusieurs conflits dans le processus de conception. Afin de faciliter la détection et le traitement des conflits, ces dépendances doivent être représentées de manière formelle. Par conséquent, à ce niveau, le chef de projet crée une catégorie de dépendance basé sur la TC afin d'illustrer les relations entre les objets. Les objets de cette catégorie font référence aux paramètres impliqués dans la conception collaborative, tandis que les morphismes représentent la dépendance entre ces paramètres. Les attributs des morphismes mettent en évidence la valeur du coefficient de dépendance entre les paramètres correspondants. Les valeurs de ces coefficients vont de 1 à 3 comme le montre le tableau (3.3). La valeur "1" fait référence à une faible dépendance entre les objets, "2" indique que les deux objets sont modérément dépendants et "3" dénote une forte dépendance. Les coefficients de dépendance sont évalués sur la base des connaissances et des expériences antérieures du chef de projet. Pour chaque système mécatronique une catégorie de dépendance unique est créée qui sera utilisée dans les étapes suivantes du processus de résolution des conflits.

TABLE 3.3 – Les différentes valeurs du coefficient de dépendance

Coefficient de dépendance	**Description des niveaux de coefficient de dépendance**
1	Dépendance faible
2	Dépendance modérée
3	Dépendance élevée

3.8.3 Étape 3 : Définition des règles de cohérences

Comme introduit précédemment, une incohérence est définie comme un état de conflit, marqué par la présence d'une contradiction. Ce terme est utilisé pour désigner une situation dans laquelle un ensemble de descriptions ne respecte pas certaines relations qui devraient exister entre elles. Ces relations peuvent être exprimées sous forme d'une règle de cohérence (CR) contre laquelle les descriptions peuvent être vérifiées [77]. Les CRs font référence à des contraintes qui doivent être respectées par les modèles impliqués dans la conception. Dans notre approche, une CR peut être écrite sous la forme suivante : CR1 = "P_{1EM2} doit être égal ou inférieur à P_{1EM1}".

3.8.4 Étape 4 : Définition des catégories de pattern *Category Pattern* (PtC)

Comme l'indique Herzig et al. [124], il est impossible de saisir toutes les incohérences en raison du manque de connaissance parfaite des processus. Par conséquent, seules les incohérences connues peuvent être définies. Ces incohérences connues peuvent être considérées comme des *patterns*. Ainsi, les catégories de pattern (PtC) visent à décrire l'existence d'un conflit dans une P_mCI (voir figure 3.15).

3.8.5 Étape 5 : Localisation des conflits

À ce niveau, les conflits présentés pendant le processus de conception peuvent être localisés en assurant un mapping entre la catégorie d'un paramètre (P_mCI) et la catégorie d'un pattern (PtC). Dans le contexte de la TC, ce mapping est établi de manière formelle par le biais de foncteurs. Comme défini précédemment, les foncteurs sont considérés comme une correspondance entre les différents objets ainsi que les morphismes d'une catégorie à une autre. Un exemple de ce mapping est illustré par la figure (3.17). Les images de y_1, y_2, m_1 et m_2 de la catégorie de pattern (PtC) à travers le foncteur "G" sont respectivement P_{1EM1}, P_{1EM2}, V_{1EM1} et V_{2EM2} dans la catégorie P_1CI. Ce mapping permet de vérifier la cohérence entre les différentes valeurs d'un même paramètre dans la catégorie P_mCI correspondante. Tout ce dont nous avons besoin est de générer un foncteur de la catégorie de pattern PtC vers la catégorie P_mCI désirée pour vérifier la cohérence. Par conséquent, si ces valeurs ne sont pas égales, un conflit est détecté et doit être géré en effectuant les étapes suivantes. Sinon, si toutes les valeurs sont cohérentes, le processus de résolution de conflit peut être arrêté à ce niveau.

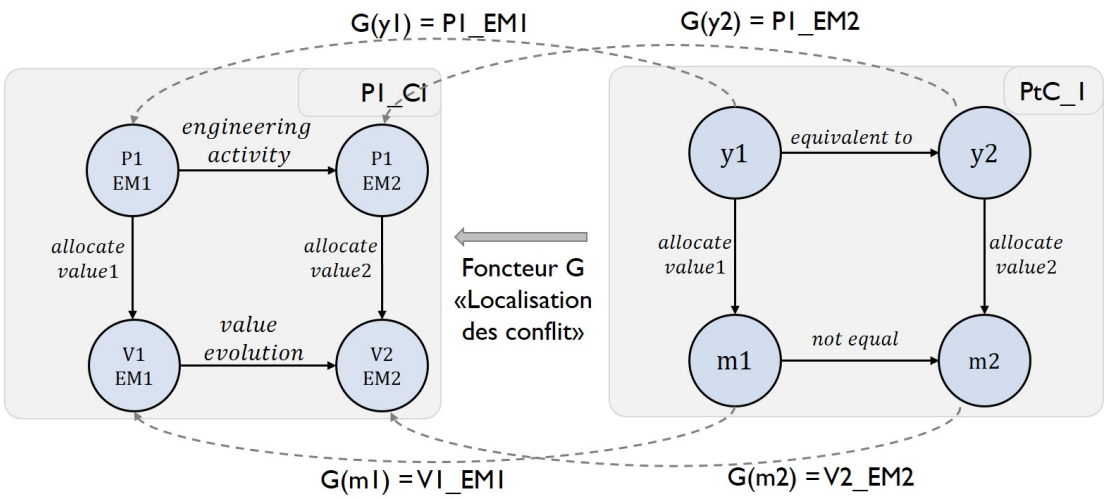


FIGURE 3.17 – Un mapping entre une catégorie d'un paramètre (P_1CI) et une catégorie de pattern (PtC_1) pour localiser les conflits

3.8.6 Étape 6 : Détection des paramètres dépendants au conflit détecté

A ce niveau, le conflit détecté doit être résolu. Si le coefficient de dépendance (c_{12}, c_{14}, c_{32}, etc. dans la figure 3.14) entre deux objets (c'est-à-dire les paramètres P_1, P_2, etc.) est égal ou inférieur à 2, le lien de dépendance est ignoré et une action de résolution (RA) doit être appliquée pour résoudre ce conflit. Dans notre approche, les actions de résolution peuvent être des actions de modification, de tolérance ou d'ignorance. Ignorer le conflit signifie que le conflit détecté n'a plus d'intérêt, tandis que tolérer les conflits fait référence à la tolérance des paramètres où le décideur (c'est-à-dire le chef de projet) décide de tolérer un paramètre en considérant un léger écart par rapport aux valeurs souhaitées. Cependant, la modification de conflit nécessite une modification de la valeur du paramètre concerné. Dans ce cas, l'ingénieur disciplinaire concerné doit mettre à jour son modèle métier jusqu'à trouver la valeur appropriée. Ainsi, l'action de résolution est basée sur un ensemble de règles de traitement, qui sont définies comme : une règle de tolérante, une règle de modificatrice ou une règle d'ignorance. Néanmoins, si le coefficient de dépendance entre deux paramètres est supérieur à 2, ce lien doit être pris en considération et l'action de résolution doit être choisie tout tenant en compte les paramètres liés au paramètre en conflit afin d'éviter l'apparition de nouveaux conflits (étape 6.2 de la figure 3.16). À ce niveau, les valeurs des paramètres connexes doivent être mises à jour et enregistrées dans l'UP_mCI (étapes 6.2.1 et 6.2.2). Ensuite, un nouveau mapping entre la PtC et la UP_mCI est établi afin de vérifier l'existence de nouveaux conflits. Si de nouveaux conflits apparaissent, l'étape 6 sera répétée jusqu'à ce que le compromis

approprié entre tous les paramètres soit atteint. Sinon, les résultats finaux seront enregistrés et le processus de résolution des conflits pourra être arrêté.

3.8.7 Étape 7 : Sauvegarde des valeurs finales obtenues après la résolution des conflits

La dernière étape de notre méthodologie CRP consiste à enregistrer les résultats finaux obtenus après la résolution de tous les conflits dans les catégories finales des paramètres FP_mCI. Cette catégorie sera utilisée pour les perspectives de traçabilité pendant le processus de collaboration.

La méthodologie CRP que nous proposons, basée sur les règles de cohérences, les patterns et les constructions mathématiques de la théorie des catégories, fournit un cadre formel pour détecter et gérer les conflits entre les différentes parties prenantes pendant la collaboration. Ce cadre est une solution flexible puisque des règles peuvent être ajoutées, modifiées ou même supprimées sans influencer tout le processus de résolution des conflits. De plus, la détection et la résolution des conflits dans notre méthodologie sont basées sur la TC, ce qui permet de conserver toutes les connaissances de la pratique d'une manière facile et formelle et facilite ainsi la traçabilité de la collaboration. Afin d'illustrer la capacité de cette approche, son application sur un boîtier papillon sera étudiée dans la section suivante.

3.9 Validation de la méthodologie CRP sur un boîtier papillon

Afin de démontrer la capacité de notre approche, le boîtier papillon décrit précédemment sera considéré comme un cas d'étude dans cette section.

3.9.1 Étape 1 : Création d'un formalisme commun pour les paramètres de l'ETB

Dans cette première étape, les modèles métiers précédemment utilisés dans la section 3.5.2 (le modèle d'exigences (EM_R), le modèle multi-physique (EM_{MP}), le modèle COTS (EM_C) et le modèle 3D (EM_{3D})) seront également employés dans cette section pour vérifier les conflits qui peuvent avoir lieu. Pour commencer alors la gestion des conflits, les catégories des paramètres ou *Parameter Category* (PC) seront instanciées. De ce fait, des valeurs seront attribuées à chaque paramètre comme le montre la figure (3.18). Cette figure présente l'instance de la catégorie de S_e. Cette instanciation s'effectue en utilisant le principe de la catégorie tranche ou *Comma*

category (définie dans la section 1.10.2). Les objets de la catégorie nommée S_e *Category Instance* sont les différentes instances du paramètre S_e (S_{eEMR} et S_{eEMMP}) ainsi que les valeurs associées à chaque instance (0,5 deg et 0,6 deg). On procède de la même manière pour tous les paramètres utilisés dans la conception de l'ETB. Le tableau (3.4) illustrent toutes les valeurs établies durant la conception collaborative de l'ETB.

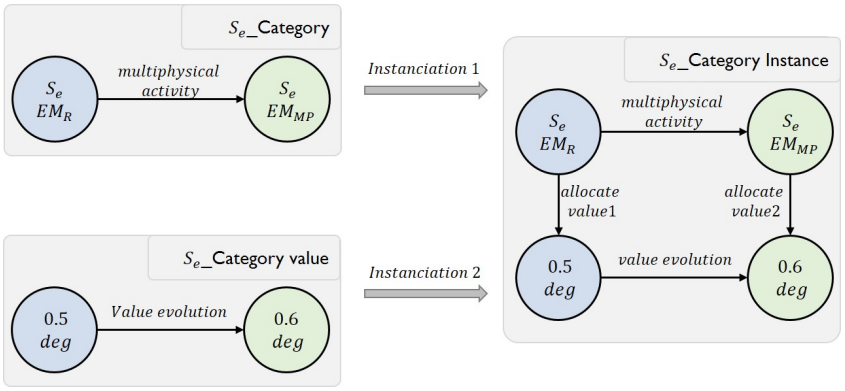


FIGURE 3.18 – Création de l'instance de la catégorie de S_e à partir de la catégorie du paramètre en se basant sur la catégorie tranche

TABLE 3.4 – Les différents paramètres utilisés dans la conception de l'ETB

Paramètres	**Symbole**	**Unité**	**EM-R**	**EM-MP**	**EM-C**	**EM-3D**
Temps de réponse	Rt	ms	200	140	-	-
Erreur statique	Se	deg	0.5	0.6	-	-
Masse globale	Mass	Kg	2.000	-	2.150	-
Puissance maximale	Pw	W	250	280	250	-
Coût	Ct	€	1500	-	1555	-
Diamètre du moteur	Ømot	mm	40	-	36	36
Longueur du moteur	$Lgth_m$	mm	60	-	75	60
Résistance du moteur	Rm	Ohm	-	4	4	-
Inductance du moteur	Lm	mH	-	1.5e-3	1.5e-3	-
Inertie du moteur	Jm	$Kg.m^2$	-	1e-6	1.5e-6	-
Diamètre du réducteur	Øred	mm	30	-	30	30
Longueur du réducteur	$Lgth_g$	mm	45	-	40	40
Ratio du réducteur	r_g	[]	-	44	44	-

3.9.2 Étape 2 : Création d'une catégorie de dépendance entre les paramètres de l'ETB

Dans la deuxième étape de notre méthodologie pour la gestion des conflits, la catégorie de dépendance de l'ETB est établie (voir figure 3.19). Le chef de projet attribue les coefficients de dépendance, indiquant le degré de dépendance par des morphismes entre les paramètres. Ces coefficients sont définis en fonction de l'expertise du chef de projet. Les actions de résolution seront ainsi choisies en fonction de coefficients de dépendance entre les paramètres, comme sera expliqué dans les étapes suivantes.

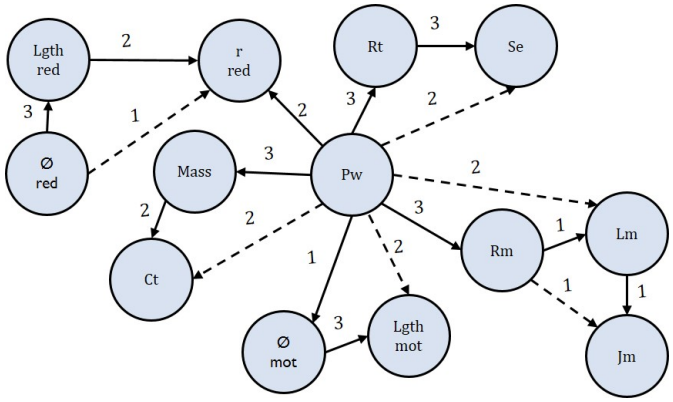


FIGURE 3.19 – Catégorie de dépendance de l'ETB, les morphismes en pointillés font référence à la composition des morphismes

3.9.3 Étape 3 : Définition des règles de cohérences pour l'ETB

Au cours de cette étape, les différentes règles de cohérences sont définies afin de décrire les relations qui doivent exister entre les différents paramètres des modèles métiers utilisés dans la conception de l'ETB. Un extrait de quelques règles de cohérences est présenté dans la liste suivante.

$CR_1 =$ " Rt_{EMMP} *doit être égal ou inférieur à* Rt_{EMR}",
$CR_2 =$ " Se_{EMMP} *doit être égale ou inférieure à* Se_{EMR}",
$CR_3 =$ " $Mass_{EMC}$ *doit être égale ou inférieure à* $Mass_{EMR}$",
$CR_4 =$ " Pw_{EMMP} *doit être égale ou inférieure à* Pw_{EMR}",
$CR_5 =$ " Pw_{EMC} *doit être égale à* Pw_{EMMP}",
.....

3.9.4 Étape 4 : Définition des catégories de pattern

En se basant sur les règles de cohérences définies précédemment, les patterns sont créés afin de formuler les conflits attendus entre les modèles métiers de l'ETB. Chaque morphisme entre deux valeurs différentes du même paramètre dans la catégorie de pattern met en évidence le non-respect d'une règle de cohérence.

3.9.5 Étape 5 : Localisation des conflits

A ce niveau, les conflits sont détectés en faisant correspondre la PtC avec la PCI de chaque paramètre crucial. Cette étape est réalisée en utilisant le concept de foncteur tel que décrit dans la section (3.8.5). Un exemple de cette mise en correspondance est illustré par la figure (3.20). Toutes les autres $PCIs$ doivent être mises en correspondance de la même manière afin de localiser les conflits survenus durant la conception de l'ETB. Les lignes grises du tableau (3.4) représentent les paramètres conflictuels détectés lors de la première itération de notre processus de résolution des conflits.

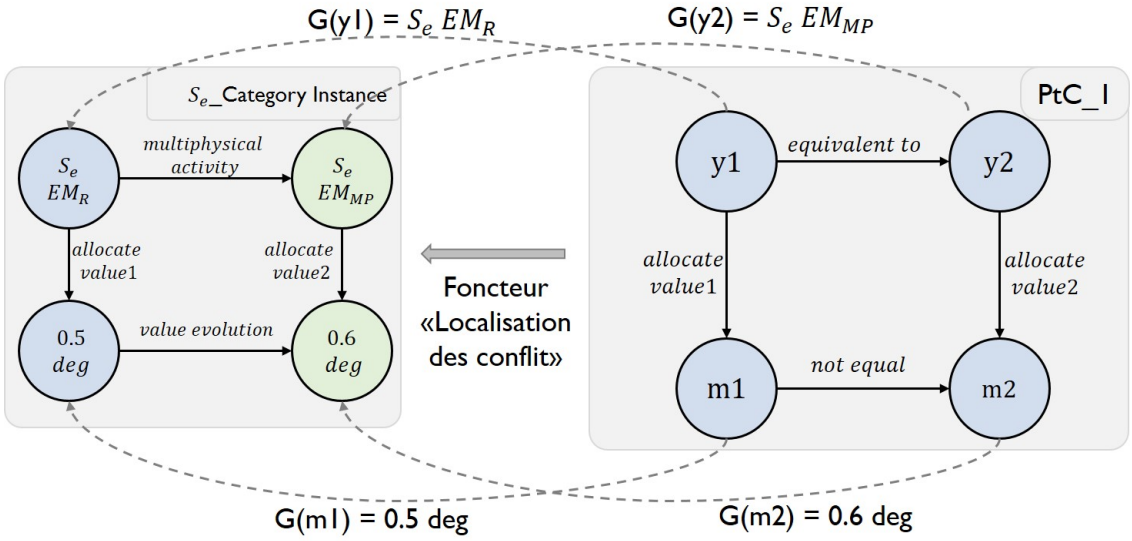


FIGURE 3.20 – Localisation du conflit dans le paramètre S_e à travers le mapping entre l'instance de S_e *Category* et la catégorie de pattern

3.9.6 Étape 6 : Détection des paramètres dépendants au conflit détecté

Une fois les conflits localisés lors de la première itération, il faut détecter les paramètres dépendants. Si le coefficient de dépendance est ≤ 2, le lien de dépendance sera ignoré. Cette décision est prise par le chef de projet afin de simplifier le processus de résolution des conflits. Dans le cas contraire, la forte dépendance entre les paramètres sera prise en compte lors de la résolution du conflit. Dans notre cas, les flux de dépendance des paramètres de la première itération sont établis en se référant à la catégorie de dépendance comme indiqué dans le tableau (3.3). A ce niveau, la valeur de l'erreur statique S_e sera modifiée, puis, les valeurs des paramètres associés (R_t, P_w, $Mass$ et R_m) seront mises à jour et sauvegardées dans les $UPCIs$ correspondantes. Après cette modification, le conflit dans la valeur de l'erreur statique (S_e) est résolu. Par conséquent, nous passons à la deuxième itération où le conflit dans la valeur de la masse sera traité. Nous suggérons dans ce cas la tolérance de ce conflit en considérant une légère modification de la valeur requise définie dans le modèle d'exigence (EM_R). Les conflits restants après cette itération sont indiqués dans la figure (3.21). Cette étape sera répétée jusqu'à ce que tous les conflits soient résolus.

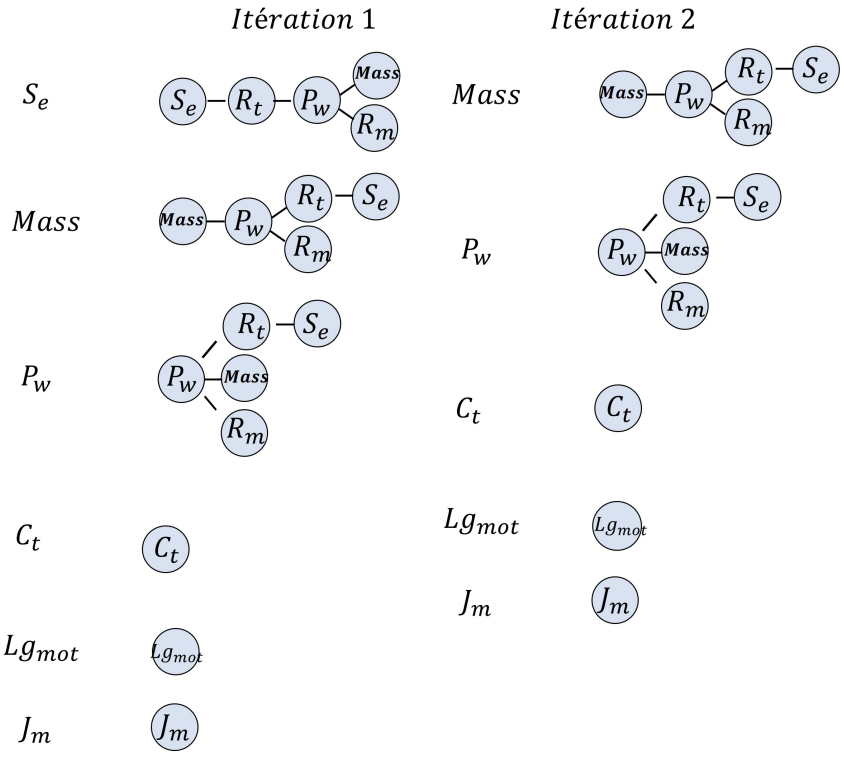


FIGURE 3.21 – Flux de dépendance entre les paramètres en conflit

Pour illustrer le principe de la gestion des conflits, nous tenons l'exemple du conflit détecté dans les valeurs de l'erreur statique. La figure (3.22) présente le scénario établi pour résoudre le conflit entre la valeur de l'erreur statique dans le modèle d'exigences et le modèle multi-physique.

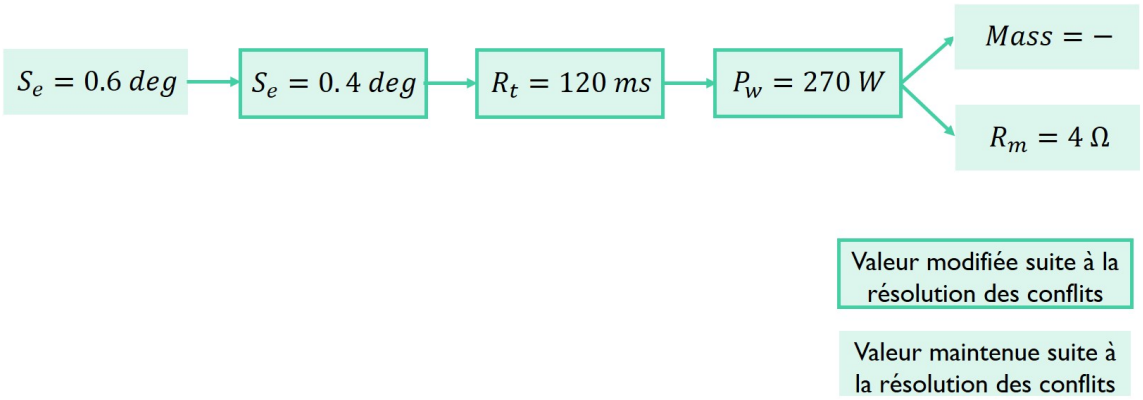


FIGURE 3.22 – Propagation de modification lors de la résolution d'un conflit entre les deux valeurs de l'erreur statique

Afin de résoudre cette incohérence, le chef de projet propose de modifier la valeur de l'erreur statique. De ce fait, le modèle multi-physique subira certaines modifications. Pour agir alors sur l'erreur statique, les paramètres du contrôleur PID seront modifiés ce qui provoquera un changement dans la valeur de l'erreur statique. Étant dépendants de l'erreur statique, le temps de réponse et la puissance maximale seront par conséquent modifiés. Cependant, la valeur de la résistance du moteur reste inchangée. Vue que cette modification a eu lieu dans le modèle multi-physique, aucune valeur pour la masse maximale n'est fournie.

Par ailleurs, pour gérer le conflit entre les deux valeurs de la masse globale dans le modèle d'exigences et le modèle COTS le chef de projet décide d'ignorer ce conflit. Cependant, le dépassement de l'exigence de coût dans le modèle COTS sera toléré en libérant cette exigence.

3.9.7 Étape 7 : Sauvegarde des valeurs finales obtenues après la résolution des conflits dans la conception de l'ETB

Une fois que tous les conflits ont été résolus avec succès, les résultats finaux sont enregistrés dans les $FPCIs$ correspondants à chaque paramètre. Le tableau (3.5) illustre les résultats obtenus après le processus de résolution des conflits. A titre d'exemple, l'instance finale de la catégorie de l'erreur statique est représentée par la figure (3.23).

TABLE 3.5 – Les résultats obtenus après la résolution des conflits dans la conception de l'ETB. Les cases colorées correspondent aux valeurs modifiées suite à l'application des règles de résolution

Paramètres	**Symbole**	**Unité**	**EM-R**	**EM-MP**	**EM-C**	**EM-3D**
Temps de réponse	Rt	ms	200	140	-	-
Erreur statique	Se	deg	0.5	0.4	-	-
Masse globale	Mass	Kg	2.000	-	2.150	-
Puissance maximale	Pw	W	250	240	250	-
Coût	Ct	€	1600	-	1555	-
Diamètre du moteur	Ømot	mm	40	-	36	36
Longueur du moteur	$Lgth_m$	mm	60	-	60	60
Résistance du moteur	Rm	Ohm	-	4	4	-
Inductance du moteur	Lm	mH	-	1.5e-3	1.5e-3	-
Inertie du moteur	Jm	$Kg.m^2$	-	1.5e-6	1.5e-6	-
Diamètre du réducteur	Øred	mm	30	-	30	30
Longueur du réducteur	$Lgth_g$	mm	45	-	40	40
Ratio du réducteur	r_g	[]	-	44	44	-

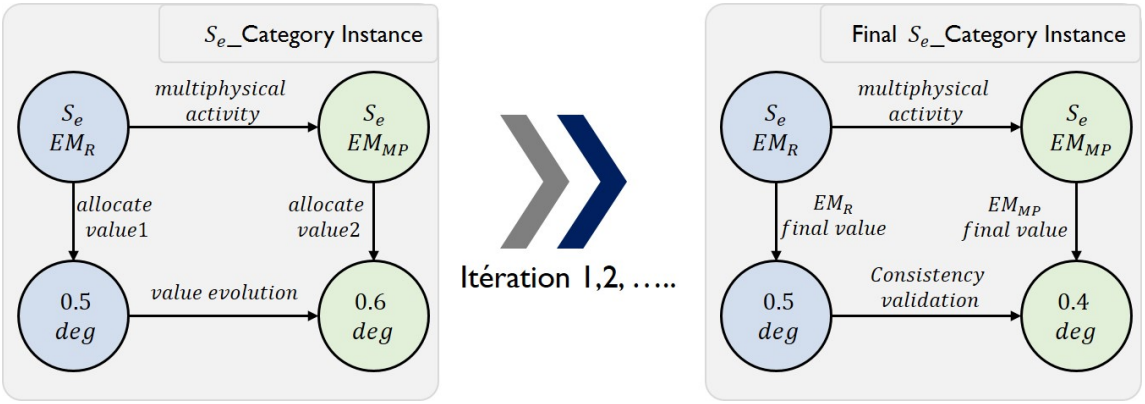


FIGURE 3.23 – FPCI de l'erreur statique

3.10 Intégration d'une méthode d'aide à la décision multicritère dans CaTCoDD

Comme nous l'avons souligné au premier chapitre, les modèles de collaboration existants dans la littérature ont démontré leur limite en termes d'aide à la décision.

De ce fait, l'objectif de cette section consiste à intégrer l'aide à la décision et le choix d'architectures dans notre modèle CaTCoDD précédemment présenté. En ce sens, nous allons proposer une nouvelle méthodologie pour le choix d'architectures que nous désignons par *Architecture Selection Process* (ASP). Cette proposition repose sur la méthode d'aide à la décision multicritère AHP (*Analytical Hierarchical Process*). Cette méthode a été choisie grâce à son modèle compréhensible et souple pour résoudre les problèmes non structurés. Pour cette raison, une description détaillée de la méthode AHP sera présentée. Les différentes étapes de notre méthodologie ASP seront également détaillées au cours de cette partie. Enfin, l'application de notre approche sur un système mécatronique aura lieu.

3.11 Description et étapes de la méthodologie Architecture Selection Process (ASP)

Dans les projets de conception complexes, les décisions sont prises en se basant sur différents critères. Chaque acteur possède des critères particuliers pour prendre une décision. Cette diversité rend la prise de décision une tâche complexe. Cette complexité ne cesse d'augmenter de plus en plus, particulièrement dans la conception des systèmes mécatroniques, où différents experts collaborent et exigent des critères qui sont parfois contradictoires. De plus, un système mécatronique peut avoir différentes architectures qui nécessitent l'évaluation afin de converger vers l'architecture la plus appropriée. Cependant, dans les supports de collaboration existants dans la littérature, cette évaluation a été négligée.

L'importance de la prise de décision a déjà été soulignée par plusieurs études [125, 126, 127]. Whelton et al. [126] affirment que près de 80% du produit et du processus sont spécifiés dans les premières phases de conception. Bellut [128] confirme également qu'une décision prise au stade préliminaire de conception aura un impact 9 fois plus important qu'une décision prise durant la phase de fabrication. Pour cela, nous proposons d'intégrer le concept d'aide à la décision et de choix d'architectures dans notre modèle de collaboration CaTCoDD. Notre approche, désignée par *Architecture Selection Process* (ASP), a pour objectif de comparer différentes architectures d'un système mécatronique dès les premières phases de conception. La structure de notre approche ASP est illustrée par la figure (3.24). L'approche ASP est composée de quatre étapes principales.

— La première étape consiste à définir les critères de comparaison.
— La deuxième étape concerne le recueil de valeurs obtenues après les processus de collaboration et gestions des conflits.
— La troisième étape de ASP se base sur la méthode AHP pour calculer le score de chaque architecture.

— Enfin, l'architecture qui possède le score le plus élevé est considérée comme la solution qui satisfait au mieux les critères définis précédemment.

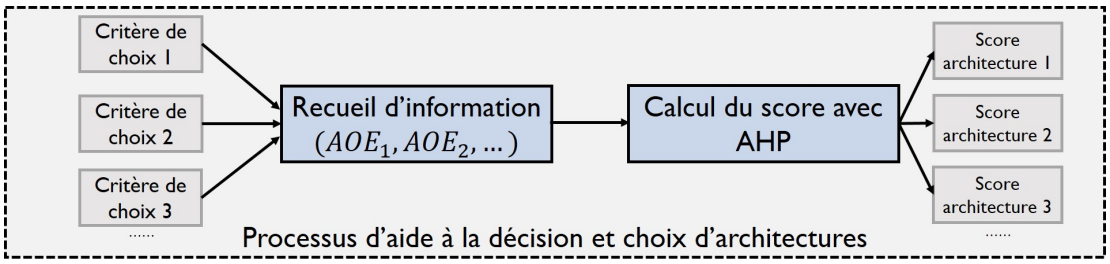


FIGURE 3.24 – Structure de la méthodologie ASP pour le choix d'architectures

3.12 Structure de la méthodologie Architecture Selection Process (ASP)

En se basant sur le processus de la méthodologie ASP présenté dans la section précédente, nous définissons le problème de choix d'architectures comme un ensemble de 3 éléments : ASP = < Criterion, Architecture Output Entity (AOE), Architecture score>.

— Le critère de choix ou *Criterion* : comme nous l'avons indiqué précédemment, le choix d'architecture se base sur un ensemble de critères. Dans notre approche, ces critères sont définis par le chef de projet en se basant sur les exigences imposées avant de commencer la conception. Il s'avère utile de préciser que le chef de projet peut comparer les différentes architectures selon l'ensemble de ces exigences ou sur un certain nombre particulier d'exigences. Ce choix revient essentiellement au degré d'importance de chaque exigence.
— L'entité de sortie de l'architecture ou *Architecture Output Entity* (AOE) : En se basant sur les critères de comparaison définis par le chef de projet, les valeurs obtenues depuis le processus de collaboration de chaque critère seront sauvegardées dans les *AOEs*. Une comparaison entre les différentes architectures sera établie selon ces valeurs.
— Le score d'architecture ou *Architecture score* : En utilisant la méthode d'aide à la décision multicritère AHP, chaque architecture aura un score précis. L'architecture avec le score le plus élevé représente la solution qui satisfait au mieux les critères de comparaison.

Vu que notre méthodologie ASP a recours à la méthode AHP, il est nécessaire de présenter les concepts fondamentaux de cette méthode.

3.13 Mise en place de la méthode Analytical Hierarchical Process (AHP) dans l'approche ASP

La méthode AHP, développée par Saaty [91], est un outil puissant de prise de décision multicritère qui a été utilisé dans de nombreuses applications dans divers domaines tels que l'économie et l'ingénierie [129]. Cette méthode consiste à attribuer une valeur qui représente le degré de préférence pour une alternative donnée à chaque alternative supplémentaire. Ces valeurs sont utilisées pour classer et sélectionner les alternatives en fonction d'une structure hiérarchique. Gupta et al. [130] affirment que la méthode AHP est la plus utilisée pour évaluer les logiciels.

Pour prendre une décision à l'aide du processus de AHP, nous devons suivre les étapes suivantes [131] :

— Étape 1 : consiste à décomposer la décision en une hiérarchie d'objectifs, de critères et d'alternatives comme le montre la figure (3.25).

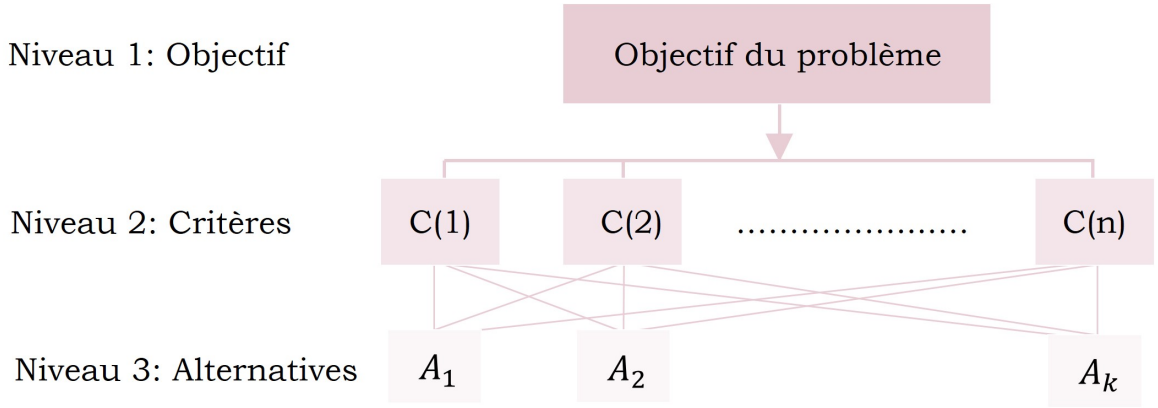


FIGURE 3.25 – Structure hiérarchique de la méthode AHP [131]

— Étape 2 : dans cette étape il est nécessaire de déterminer les priorités (pondérations) des critères. L'importance des critères est comparée par paires par rapport à l'objectif souhaité afin d'en déduire leur pondération sous forme d'une matrice nommée matrice de jugement. La comparaison s'effectue en se basant sur l'échelle de Saaty [91] présentée dans la section (2.9.4). La matrice de jugement se présente de la façon suivante :

$$A = [a_{i,j}] = \begin{pmatrix} 1 & a_{1,2} & \cdots & a_{1,n} \\ 1/a_{1,2} & \cdots & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ 1/a_{1,n} & 1/a_{2,n} & \cdots & 1 \end{pmatrix} \tag{3.1}$$

où n représente le nombre de critères de comparaison.

La cohérence des jugements est ensuite vérifiée afin d'assurer un niveau raisonnable de cohérence en termes de proportionnalité et de transitivité. Pour cela, le rapport de cohérence ou *Consistency ratio* (Cr) est calculé en comparant l'indice de cohérence ou *Consistency index* (CI) de la matrice qui contient nos jugements à l'indice de cohérence moyen ou *Random Index* (RI). Une matrice aléatoire est une matrice dans laquelle les jugements ont été saisis de manière aléatoire et qui est donc censée être très incohérente. Cet indice peut être défini comme l'IC moyen de 500 matrices remplies au hasard. Saaty [91] fournit la valeur RI calculée pour des matrices de différentes tailles, comme indiqué dans le tableau (3.6). Le rapport de cohérence (Cr) peut être ainsi calculé par l'expression suivante :

$$Cr = CI/RI \tag{3.2}$$

où l'indice de cohérence (CI) est exprimé de la façon suivante :

$$CI = (\lambda max - n)/(n-1) \tag{3.3}$$

λmax est la valeur propre maximale correspondante à la matrice des comparaisons par paires et n représente le nombre de critères.
Selon Saaty, un Cr égal à 0.10 ou moins est acceptable pour poursuivre l'analyse. Cependant, si le rapport de cohérence dépasse 0.10, il est nécessaire de réviser les jugements afin de réduire les incohérences.

TABLE 3.6 – Indices de cohérence pour une matrice générée de manière aléatoire [91]

n	3	4	5	6	7	8	9	10
RI	0.58	0.9	1.12	1.24	1.32	1.41	1.45	1.49

— Étape 3 : consiste à déterminer les priorités (préférences) locales pour les différentes alternatives en les comparant deux à deux par rapport à chaque critère.
— Étape 4 : à ce niveau, les priorités globales sont identifiées. Les priorités obtenues pour les alternatives sont combinées sous forme de somme pondérée tout en tenant compte du poids de chaque critère.
— Étape 5 : enfin, une décision finale peut être prise sur la base des résultats obtenus tout au long de ce processus. L'alternative ayant la priorité globale la plus élevée constitue le meilleur choix.

3.14 Validation de la méthodologie ASP sur un boîtier papillon

Afin de mieux illustrer le concept de la méthodologie ASP, le boîtier papillon sera utilisé pour la validation de notre approche. Dans cette validation, deux architectures différentes de l'ETB seront traitées. La première architecture est un boîtier papillon avec un moteur DC et la deuxième architecture sera avec un moteur pas à pas.

3.14.1 Étape 1 : Identification des critères de comparaison

Dans la conception de l'ETB, trois critères de comparaison sont identifiés : le temps de réponse, l'erreur statique et la masse du moteur avec le réducteur. En se basant sur ces critères, la comparaison entre les deux architectures de l'ETB sera effectuée.

3.14.2 Étape 2 : Collecte de données relatives à chaque architecture

A ce niveau, la valeur de chaque critère de comparaison est identifiée. Le tableau (3.7) illustre les différentes valeurs de l'erreur statique, le temps de réponse et la masse pour l'ETB avec un moteur DC et un moteur pas à pas [58].

TABLE 3.7 – Les valeurs de chaque critère de comparaison

Critères	**Architecture 1 (Avec moteur DC)**	**Architecture 2 (Avec moteur pas à pas**
Temps de réponse (Tr)	150 ms	280 ms
Masse (M)	210 g	90 g
Erreur statique (Es)	0.6 deg	0.5 deg

3.14.3 Étape 3 :Calcul des scores pour chaque alternative en se basant sur la méthode AHP

Dans cette étape, nous disposons de toutes les valeurs nécessaires pour choisir la meilleure architecture de l'ETB. Nous commençons par créer la matrice de jugement représentée par le tableau (3.8) :

TABLE 3.8 – Matrice de jugement dans la conception de l'ETB

	Tr	M	Es
Tr	1	5	3
M	1/5=0.2	1	1/3=0.333
Es	1/3=0.333	3	1
Somme	1.533	9	4.333

Ensuite, cette matrice nécessite la normalisation à travers la division de chaque case par le total de la colonne comme représenté par le tableau (3.9) :

TABLE 3.9 – Matrice normalisée

	Tr	M	Es	Priorités
Tr	0.652	0.555	0.692	0.599
M	0.130	0.111	0.076	0.105
Es	0.217	0.333	0.230	0.260

À partir de cette matrice normalisée, nous obtenons les priorités (pondération des critères) en calculant simplement la valeur moyenne de chaque ligne comme nous pouvons le voir sur la dernière colonne du tableau (3.9). D'après ces résultats, nous constatons que nous accordons plus d'importance au temps de réponse (0.599), suivi de l'erreur statique. Cependant, la masse a un poids minimum (0.105) dans notre choix entre les deux architectures de l'ETB.

Une fois les différentes priorités (pondérations des critères) obtenues, il est important de vérifier la cohérence de nos jugements. Pour cela nous devons calculer le rapport de cohérence (Cr) à travers l'indice de cohérence (CI). Nous commençons par multiplier chaque valeur de la première colonne de la matrice de comparaison (tableau 3.8) par la priorité du premier critère (c'est-à-dire 1*0.599 = 0.599 ; 0.130*0,599 = 0.525), multiplier chaque valeur de la deuxième colonne par la priorité du deuxième critère, etc. Ensuite, nous additionnons les valeurs de chaque ligne

pour obtenir un ensemble de valeurs appelé somme pondérée, comme indiqué dans le tableau (3.10).

TABLE 3.10 – Calcul des colonnes pondérées et de la somme pondérée

	Tr	M	Es	Somme pondérée
Tr	0.599	0.525	0.780	1.904
M	0.119	0.105	0.086	0.310
Es	0.199	0.315	0.260	0.774

Enfin, nous divisons les éléments du vecteur de la somme pondérée par la priorité correspondante de chaque critère, comme indiqué dans le tableau (3.11). La moyenne de ces valeurs correspond à λmax.

TABLE 3.11 – Calcul de λmax

Somme pondérée	Priorité	
1.904/	0.599	3.170
0.310/	0.105	2.952
0.774/	0.260	2.976
	λmax	3.035

A ce niveau, nous pouvons calculer la valeur de CI comme suit, où n=3 :

$$CI = (\lambda max - n)/(n-1) = (3.035 - 3)/(3-1) = 0.0175 \tag{3.4}$$

La valeur de CI nous permet de calculer la valeur de Cr de la façon suivante, où RI= 0.58 d'après le tableau (3.6) :

$$Cr = CI/RI = 0.0175/0.58 = 0.030 \tag{3.5}$$

Nous pouvons remarquer que le rapport de cohérence (Cr) de notre matrice de jugement est inférieur à 0,10. Par conséquent, notre matrice est raisonnablement cohérente et nous pouvons continuer le processus de prise de décision.

Pour continuer le processus de décision, nous passons au calcul des priorités relatives des alternatives par rapport à chaque critère. Pour cela, une comparaison par paire de toutes les alternatives par rapport à chaque critère est nécessaire (en se basant sur l'échelle représentée par le tableau 2.2 dans la section 2.9.4). Les tableaux

(3.12), (3.13) et (3.14) illustrent la comparaison entre les deux architectures de l'ETB par rapport au temps de réponse, la masse et l'erreur statique respectivement.

TABLE 3.12 – Priorités obtenues par rapport le temps de réponse

Tr	A1	A2	Priorité
A1	1	7	0.875
A2	1/7= 0.143	1	0.125
Somme	1.143	8	

TABLE 3.13 – Priorités obtenues par rapport la masse

M	A1	A2	Priorité
A1	1	1/9=0.111	0.100
A2	9	1	0.900
Somme	10	1.111	

TABLE 3.14 – Priorités obtenues par rapport l'erreur statique

Es	A1	A2	Priorité
A1	1	1/5=0.200	0.167
A2	5	1	0.833
Somme	6	1.200	

Ces résultats nous permettent de dériver les priorités locales des architectures par rapport à chaque critère comme le montre le tableau (3.15).

En se basant sur la priorité locale de chaque alternative calculée précédemment, nous allons calculer la priorité globale. Nous allons également prendre en considération les poids de chaque critère calculés (pondérations des critères) dans le tableau (3.9). Ce calcul s'effectue par la multiplication de chaque priorité locale par le poids du critère correspondant et en les additionnant. Les priorités globales de chaque architecture de l'ETB sont représentées par le tableau (3.16)

TABLE 3.15 – Priorités locales des architectures de l'ETB par rapport à chaque critère

Alternatives	Tr	M	Es
Architecture 1	0.875	0.1	0.167
Architecture 2	0.125	0.9	0.833

TABLE 3.16 – Priorités globales des architectures

	Tr	M	Es	Priorité globale
Pondération des critères	0.599	0.105	0.260	
Architecture 1	0.875	0.1	0.167	0.578
Architecture 2	0.125	0.9	0.833	0.385

3.14.4 Étape 4 : Choix d'architecture

En se basant sur les scores obtenus pour chaque architecture de l'ETB, nous pouvons constater que l'architecture 1 (avec le moteur DC) possède une priorité globale plus importante que la deuxième architecture (avec le moteur pas à pas). De ce fait, cette architecture répond le mieux aux critères et elle est considérée comme l'architecture la plus appropriée.

3.15 Conclusion

Dans ce chapitre, nous nous sommes intéressés à l'identification des connaissances cruciales dans la conception des systèmes mécatroniques. En ce sens, nous avons proposé une nouvelle méthodologie que nous désignons par *Crucial Knowledge Identification Process* (CKIP). Cette méthodologie a permis l'identification des connaissances cruciales pendant un processus de conception collaborative mécatronique en se basant sur la théorie des catégories. La méthodologie CKIP a fourni une représentation formelle et unifiée des connaissances impliquées dans la collaboration. L'architecture ainsi que les différentes étapes de la méthodologie proposée ont été soulignées dans ce chapitre. Une comparaison entre la méthodologie CKIP et les méthodologies existantes dans la littérature a été effectuée au cours de ce chapitre. Cette comparaison a principalement pour but de montrer notre contribu-

tion en termes d'identification formelle des connaissances cruciales. En se basant sur les connaissances cruciales extraites à travers la méthodologie CKIP, une deuxième méthodologie pour la localisation et la gestion des conflits a été développée. Notre méthodologie nommée *Conflict Resolution Process* (CRP) et basée sur la théorie des catégories, a fourni une méthode formelle de détection des conflits basée sur le mapping entre les différentes instances des catégories des paramètres et les catégories des patterns au moyen de foncteurs. Le formalisme de représentation commun basé sur les concepts de la TC a permis également la traçabilité de la collaboration, ce qui est efficace pour les perspectives de réutilisation. Ces deux propositions ont été appliquées à un boîtier papillon permettant le réglage du débit d'air dans le moteur d'automobile. Une seule architecture a été prise en compte dans ces méthodologies. Cependant comme nous l'avons déjà indiqué, chaque système mécatronique peut avoir différentes architectures. Pour cela, l'aide à la décision et le choix d'architectures ont fait l'objet de la dernière partie de ce chapitre. Notre méthodologie désignée par *Architecture Selection Process* (ASP) a été développée. Cette méthode a recours à la méthode AHP pour calculer le score de chaque architecture afin d'aider le chef de projet à choisir l'architecture qui satisfait au mieux les exigences. Notre approche ASP a été également appliquée au boîtier papillon en comparant deux architectures différentes. L'approche ASP nous a permis de conclure que l'architecture de l'ETB avec un moteur DC répond le mieux aux critères de comparaison.

Après avoir détaillé les différents axes de recherche de notre méthodologie générale CaTCoDD, nous allons illustrer son efficacité à travers son application à un actionneur électromécanique d'aileron dans le chapitre suivant. Notre démonstrateur CaTCoDD sera utilisé dans cette validation pour mieux illustrer les concepts de nos propositions.

Chapitre 4

Validation de la méthodologie CaTCoDD

Ce chapitre valide notre méthodologie CaTCoDD en utilisant l'exemple de l'actionneur électromécanique d'aileron (*Electro-Mechanical Actuator of an aileron* EMA). Des modèles métiers multidisciplinaires sont présentés au cours de cette validation. Le scénario collaboratif de la conception de l'EMA sera appliqué à chaque étape de notre méthodologie générale. Le démonstrateur décrit précédemment sera utilisé dans ce chapitre afin de présenter une vision réelle de l'implémentation de notre méthodologie.

4.1 Description de l'actionneur Électromécanique d'aileron (EMA)

L'utilisation des actionneurs électromécaniques dans le domaine aéronautique a fait l'objet de nombreuses études antérieures [132, 133, 134]. En les comparant avec les actionneurs hydrauliques, les actionneurs électromécaniques, remplaçant l'hydraulique par une chaîne de transmission de puissance mécanique, ont montrés leur efficacité en termes de réduction significative de la maintenance en raison d'une moindre conversion d'énergie [135]. L'utilisation des ces actionneurs dans les commandes de vol est de plus en plus fréquente car ils présentent de nombreux avantages. La figure (4.1) montre les différentes surfaces de contrôle de vol de la famille Airbus A320 [136].

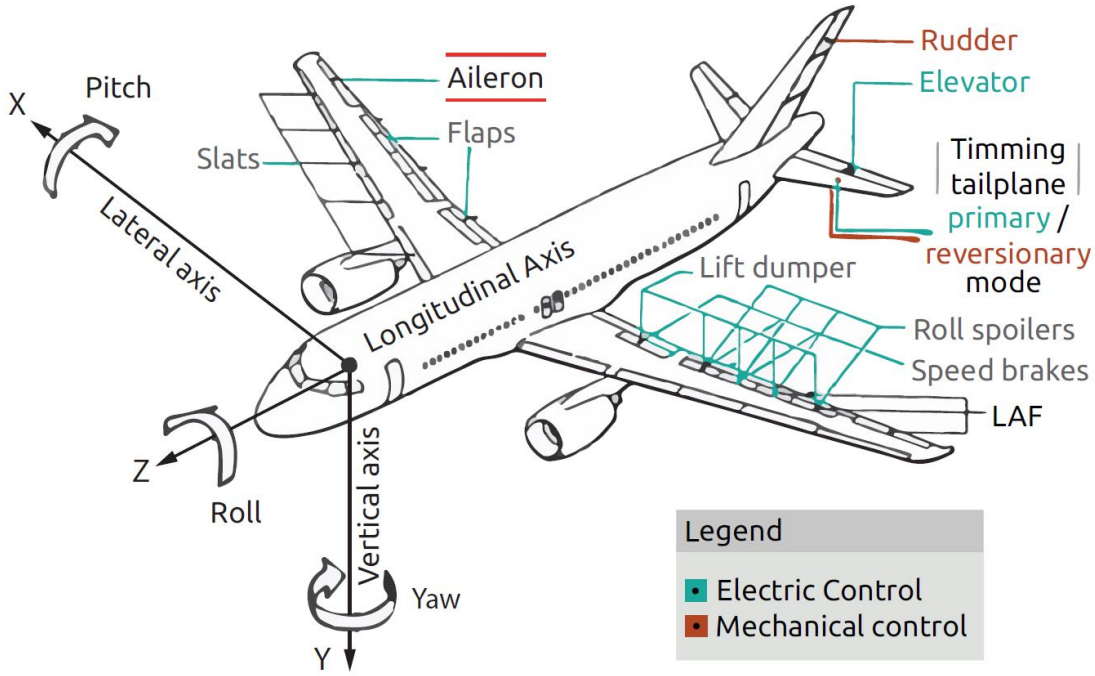


FIGURE 4.1 – Surfaces de contrôle de vol de la famille Airbus A320 [136]

Mami [137] affirme dans son étude que l'EMA présente un meilleur respect de l'environnement avec la suppression de la puissance hydraulique et des risques de fuite d'huile, un gain de poids sur l'avion ainsi qu'une augmentation des performances et précision de la vitesse. L'EMA est constitué de trois parties principales : une partie électrique composée d'un moteur électrique, une partie mécanique formée par la transmission et une partie électronique et logicielle composée d'un calculateur qui contrôle le système [138].

L'objectif principal de l'EMA consiste à actionner d'aileron de l'avion et remplacer les mécanismes habituels à tige, levier et câbles [139]. La structure générale de l'EMA est représentée par la figure (4.2).

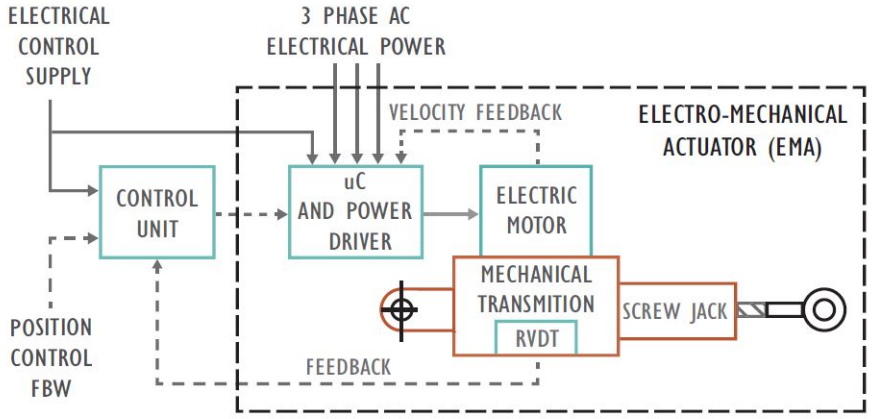


FIGURE 4.2 – Structure de l'actionneur électromécanique de d'aileron [136]

Plusieurs architectures de l'EMA peuvent être étudiées telles qu'une architecture à entraînement direct, à 3 barres ou à 4 barres. Cet actionneur est considéré comme un système multi-physique puisqu'il englobe la mécanique, la commande et l'électrique. Son aspect pluridisciplinaire fait de l'EMA un exemple intéressant pour valider notre approche. Les différentes étapes de notre méthodologie CaTCoDD décrites précédemment seront appliquées à l'EMA.

4.2 Étape 1 : Initialisation

C'est une étape de pré-collaboration où les éléments nécessaires à la collaboration sont établis.

4.2.1 Spécification des exigences

Le chef de projet commence le processus de conception par définir les différentes exigences qui doivent être respectées. Notre objectif consiste à concevoir un EMA robuste et cohérent en respectant les exigences suivantes :

— Le temps de réponse de l'EMA ne doit pas dépasser 600 ms
— L'erreur statique doit être inférieure à 2 degrés
— La masse de l'EMA est de 3 Kg
— La puissance ne doit pas dépasser 250 W
— Le coût du système ne doit pas dépasser 2000 €

— Les composants de l'EMA doivent être intégrés dans la zone allouée entre d'aileron et l'aile.

4.2.2 Identification d'architectures de l'EMA

Comme nous l'avons déjà souligné, différentes architectures de l'EMA peuvent se présenter. Nous avons choisi dans notre validation deux architectures de l'EMA pour illustrer notre approche, à savoir l'architecture à 3 barres et l'architecture à entraînement direct. Ces architectures sont illustrées dans la figure (4.3). La proposition de ces architectures s'effectue par l'ingénieur système. Pour mieux illustrer la différence entre ces deux architectures, les diagrammes de structure pour chaque architecture sont représentés par les figures (4.4) et (4.5). Ces diagrammes sont des diagrammes de définition de bloc ou *Bloc Definition Diagram* (BDD) créés avec le langage SysML dans l'environnement MagicDraw.

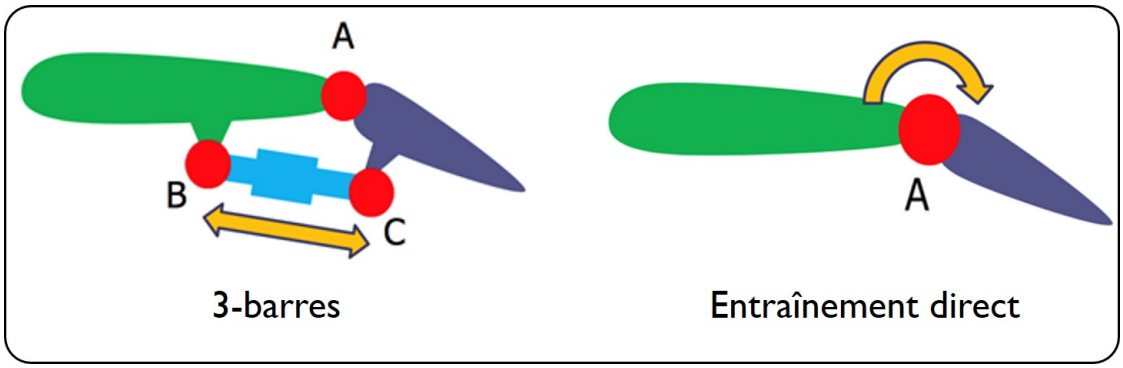


FIGURE 4.3 – Les deux architectures de l'EMA à comparer [139]

4.2.3 Identification des modèles métiers

Afin de valider notre méthodologie CaTCoDD, nous avons créé quatre modèles métiers utilisés dans la conception collaborative de l'EMA. Le premier modèle est un modèle d'exigences (EM_R). Le deuxième représente un modèle multi-physique (EM_{MP}) et le troisième illustre le choix des composants (EM_C). Le dernier modèle est un modèle 3D (EM_{3D}). les modèles disciplinaires de l'EMA seront détaillés ci-dessous.

Modèle d'exigences

Le modèle d'exigences (EM_R) a pour objectif d'identifier les exigences qui doivent être respectées durant la conception de l'EMA. Ce modèle est créé à l'aide du langage

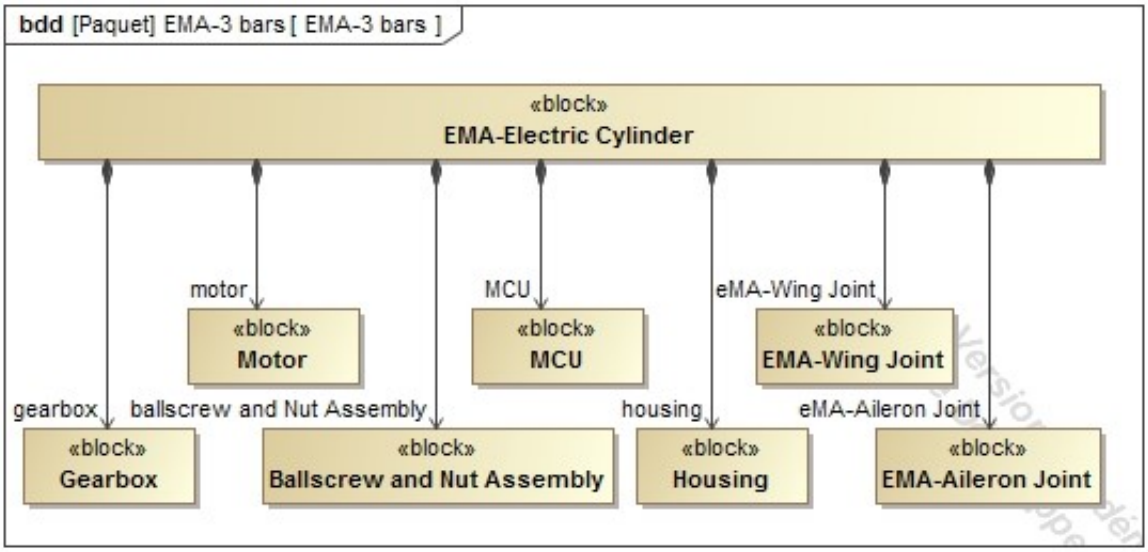


FIGURE 4.4 – Diagramme BDD de l'architecture de l'EMA avec 3 barres

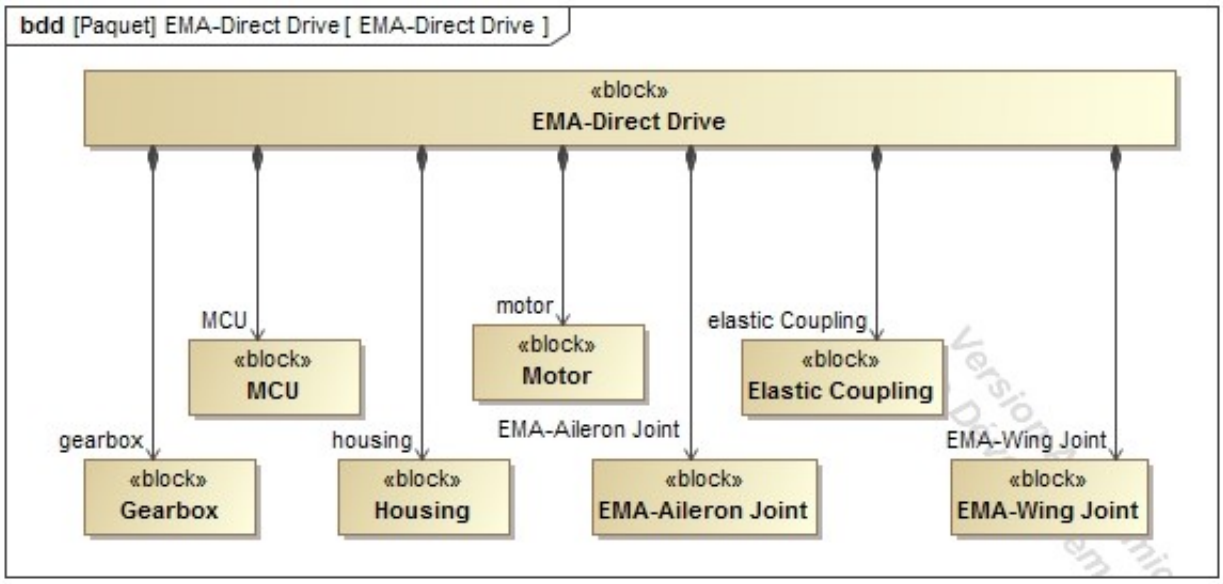


FIGURE 4.5 – Diagramme BDD de l'architecture de l'EMA avec entraînement direct

SysML comme indiqué dans la figure (4.6).

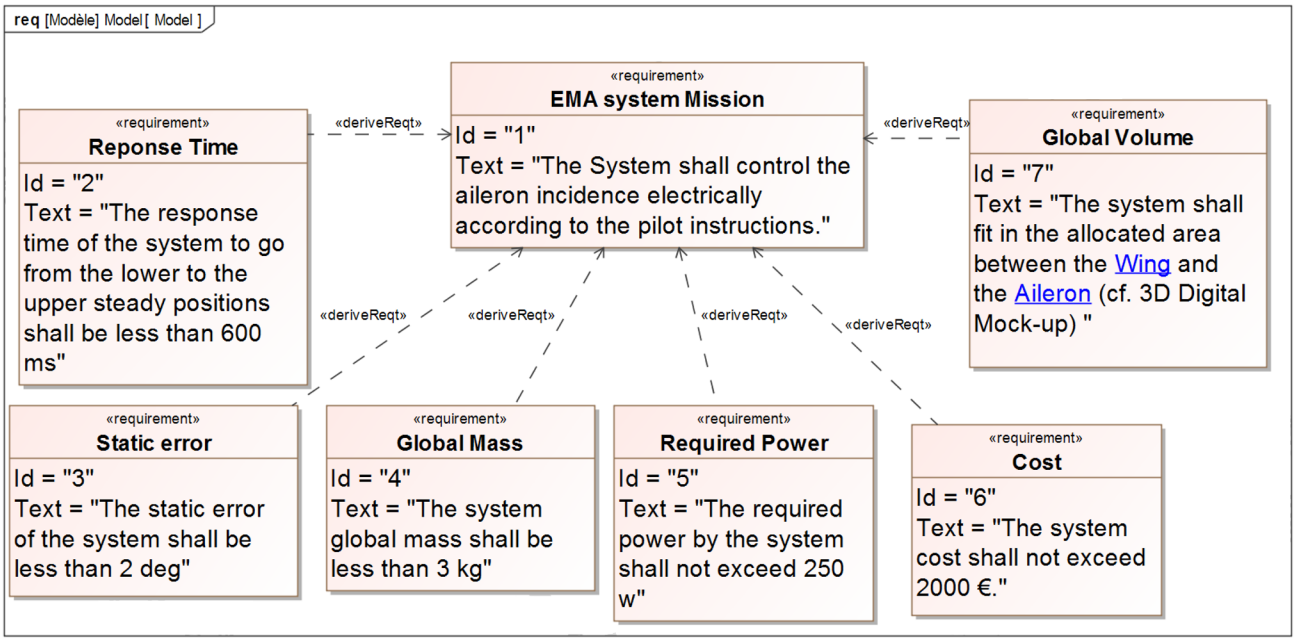


FIGURE 4.6 – Exigences initiales de l'EMA

Modèle multi-physique

Le deuxième modèle (EM_{MP}) est un modèle multi-physique créé avec le langage Modelica. Ce modèle est basé sur des équations différentielles, discrètes et algébriques afin de décrire la réponse dynamique du système [140]. En utilisant le langage Modelica, il est possible de simuler des prototypes virtuels et d'évaluer le comportement physique des systèmes multidisciplinaires, ce qui est adapté à la conception des systèmes mécatroniques [56].

Vu que nous étudions deux architectures différentes de l'EMA, deux modèles multi-physiques se présentent. La figure (4.7) illustre le modèle Dymola de l'architecture à 3 barres. Ce modèle est créé en utilisant des composants de la bibliothèque Modelica. Il contient un moteur à courant continu alimenté par un contrôleur PID et connecté à un réducteur. Le mouvement de rotation produit est converti en translation au moyen d'une vis-écrou. Un couple aérodynamique a été également appliqué comme nous pouvons le constater sur la figure (4.7). Ce couple est proportionnel au sinus carré de l'angle entre l'aileron et l'horizontal [141]. Nous avons modélisé le couple aérodynamique sous forme d'un bloc dans le modèle Dymola. La structure de ce bloc est illustrée par la figure (4.8).

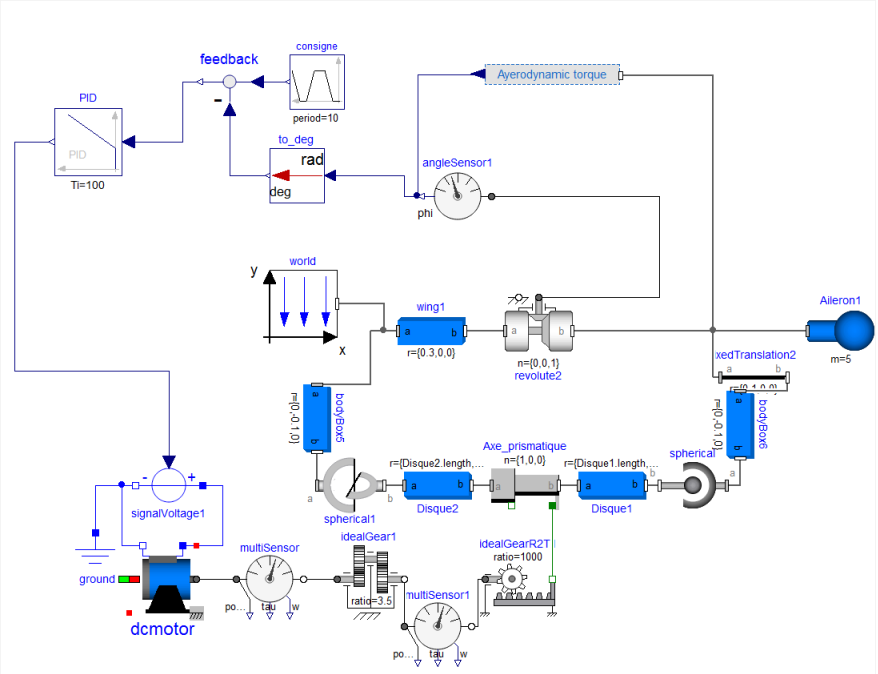


FIGURE 4.7 – Modèle multi-physique de l'EMA avec 3 barres dans l'environnement Dymola

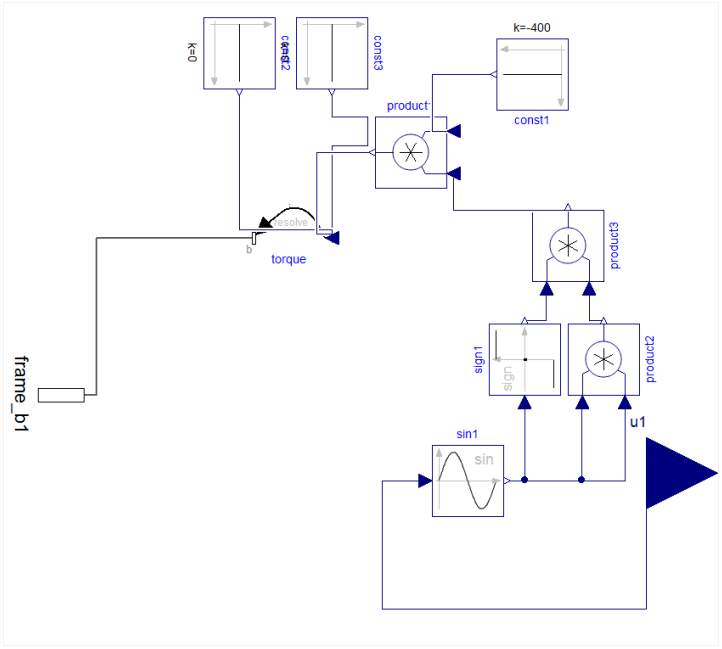


FIGURE 4.8 – Couple aérodynamique appliqué à l'EMA dans le modèle Dymola

La deuxième architecture de l'EMA est une architecture à entraînement direct ou *Direct drive* comme représenté par la figure (4.9). Cette architecture est composée d'un moteur alimenté par un courant continu. Ce moteur alimente à son tour la liaison pivot après la réduction (représentée par le composant *revolute* dans le modèle Dymola). Cette liaison est reliée directement à l'aileron. La sortie de ce modèle sera la position angulaire de la liaison pivot. Le couple aérodynamique (présenté ci-dessus) sera également appliqué au modèle de l'EMA avec entraînement direct.

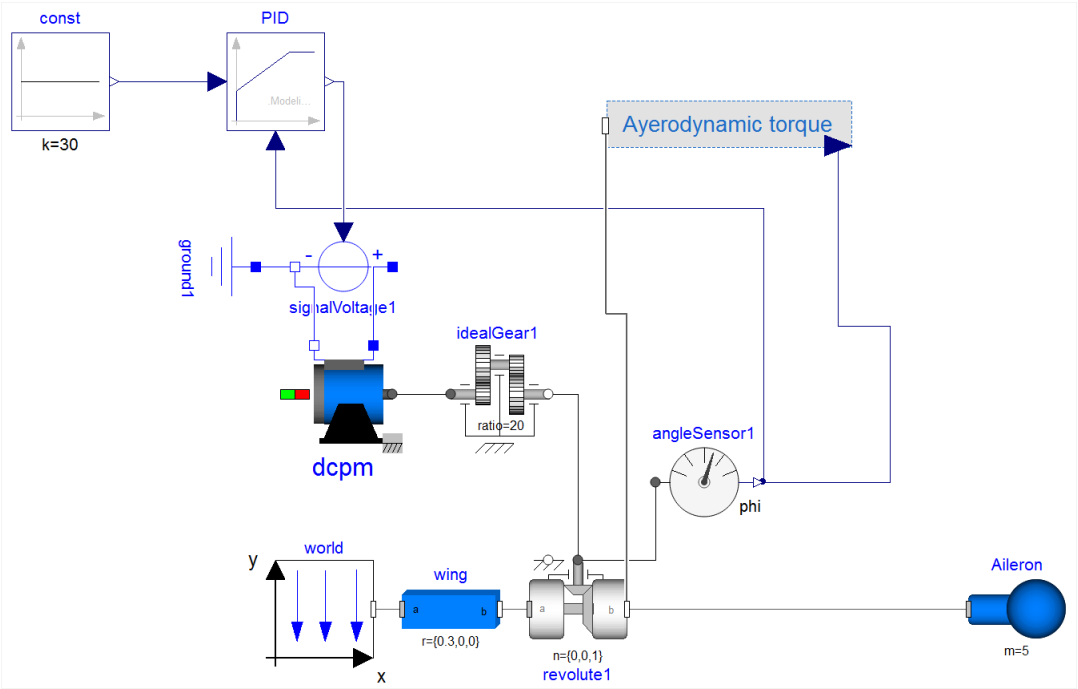


FIGURE 4.9 – Modèle multi-physique de l'EMA avec entraînement direct dans l'environnement Dymola

Modèle COTS

Sur la base du modèle multi-physique et les résultats de simulation obtenues, un autre modèle métier est établi (EM_C). Les composants commerciaux disponibles sur étagère ou *Commercial Off-The-Shelf* (COTS) sont choisis en fonction des résultats de la simulation. Pour l'architecture à 3 barres, un moteur DC, un réducteur et une vis-écrou sont choisis. Cependant dans la deuxième architecture uniquement un moteur et un réducteur seront sélectionnés. Un extrait de ces composants est illustré par la figure (4.10). Les caractéristiques du moteur sélectionné sont disponibles en annexe 2.

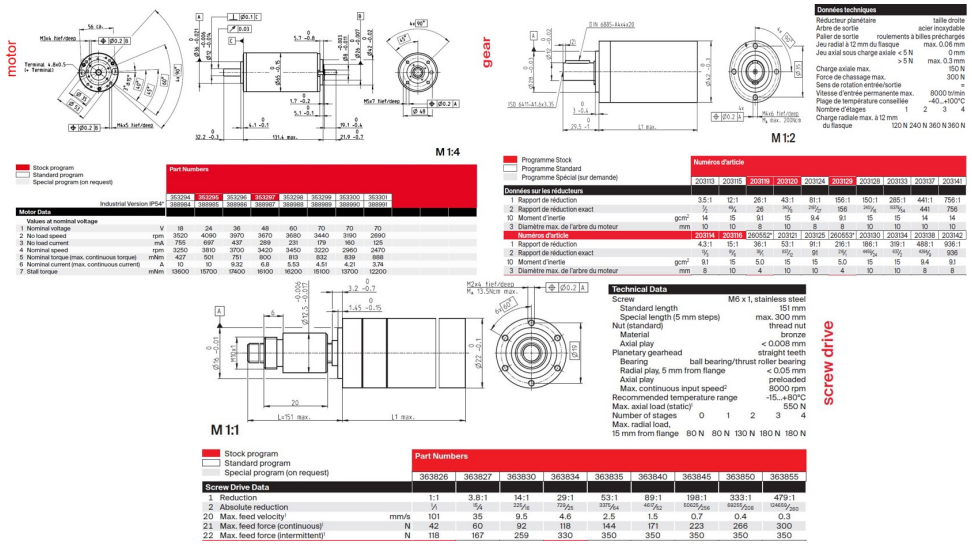


FIGURE 4.10 – Caractéristiques des composants extraites de [119]

Modèle 3D

Le dernier modèle métier qui sera utilisé dans la conception de l'EMA est un modèle 3D (EM_{3D}) en utilisant l'environnement CATIA. Ce modèle est créé afin de vérifier l'intégration de l'ensemble du mécanisme dans l'assemblage de l'aileron et l'aile. Les figures (4.11).a et (4.11).b représentent le modèle 3D pour l'architecture à 3 barres et l'architecture avec un entraînement direct respectivement.

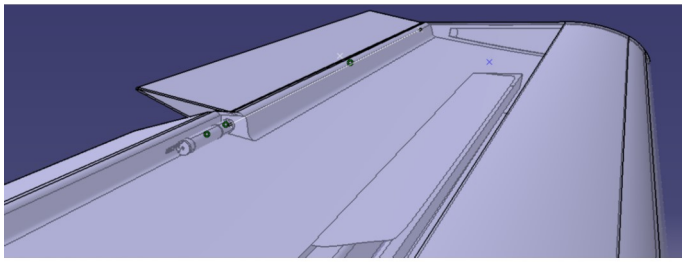
4.2.4 Identification du workflow

Bien que nous parlions de la conception collaborative, il y a toujours un *workflow* bien défini lors de la création de chaque modèle métier. Dans notre étude de cas, le modèle d'exigences (EM_R) est fourni au début. Ensuite, le modèle multi-physique (EM_{MP}) de l'EMA est présenté pour dimensionner les composants afin de satisfaire les exigences identifiées. Une fois les résultats de simulation obtenus à partir du modèle mylti-physique, les composants COTS existants (EM_C) doivent être sélectionnés. Le dernier modèle (EM_{3D}) consiste à vérifier l'intégration 3D de l'ensemble du mécanisme dans l'assemblage de l'aile et de l'aileron.

4.3 Étape 2 : Collaboration

C'est une étape intermédiaire dans notre méthodologie. Les ingénieurs disciplinaires interviennent à ce niveau pour collaborer ensemble afin de partager leurs

(a)



(b)

FIGURE 4.11 – Modèles 3D de l'architecture à 3 barres et à entraînement direct [139]

connaissances et assurer la cohérence du système mécatronique. Cette étape contient à son tour quatre sous-étapes qui seront détaillées dans ce qui suit.

4.3.1 Identification des connaissances cruciales

Comme nous l'avons déjà souligné tout au long de ce manuscrit, uniquement les connaissances qui ont réellement un rôle dans la collaboration entre les différents ingénieurs disciplinaires doivent être partagées et capitalisées. Pour cela, l'extraction des connaissances cruciales est considérée comme une étape primordiale pour la réussite de la collaboration. De ce fait, notre méthodologie CKIP proposée dans le chapitre précèdent sera appliquée à l'EMA afin d'identifier les connaissances cruciales. Notre méthodologie CKIP commence par la création des catégories pour chaque paramètre spécifié dans les exigences. Ces paramètres sont le temps de réponse (Rt), l'erreur statique (Se), masse globale ($Mass$), puissance (Pm), coût (Ct), diamètre moteur (Ømot), longueur moteur (L_{mot}), diamètre réducteur (Øred), longueur réducteur (L_{red}), longueur vis-écrou (L_{sn}) et diamètre vis-écrou (Øsn) (la vis-écrou pour l'architecture à 3 barres uniquement). Chaque catégorie représente un paramètre particulier et sera ensuite remplie par un ensemble d'objets pour représenter les différentes instances de ce paramètre. Chaque objet aura un morphisme d'identité.

Dans la conception de l'EMA, les ingénieurs disciplinaires ont besoin d'autres paramètres qui ne sont pas définis dans les exigences. De ce fait, des nouvelles catégories représentant les paramètres nécessaires sont créées. Ces paramètres sont la résistance du moteur (R_m), inductance du moteur (L_m), inertie du moteur (J_m), ratio du réducteur (r_{red}), ratio de la vis-écrou (r_{sn}) (uniquement pour l'architecture à 3 barres), constante du moteur (K_m), rendement du moteur (Eff_{mot}), rendement du réducteur (Eff_{red}), jeu moyen à vide du réducteur (P_{red}), jeu radial du moteur (Pr_{mot}) et axe de rotation pour la liaison pivot (Ax). Chaque paramètre sera représenté par une catégorie où les objets initiaux ainsi que leurs morphismes d'identité seront créés.

La dernière étape de notre approche CKIP consiste à représenter l'instanciation d'un paramètre particulier par différentes activités d'ingénierie. En ce sens, chaque catégorie sera enrichie par différents objets qui représentent les différentes instances du paramètre. Ces objets sont reliés entre eux avec des morphismes représentant les activités d'ingénierie. La catégorie de la puissance maximale est illustrée par la figure (4.12) dans notre démonstrateur CaTCoDD.

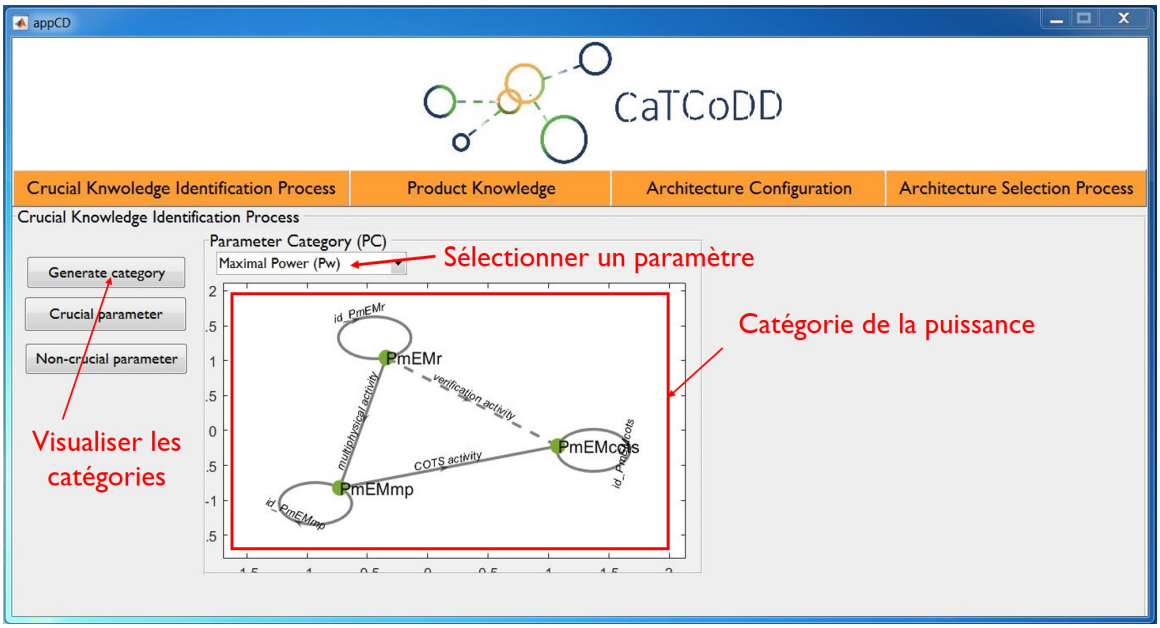


FIGURE 4.12 – Catégorie de la puissance dans le démonstrateur CaTCoDD

Nous pouvons remarquer que la catégorie de la puissance est constituée de trois objets. Le premier objet représente la puissance exigée par le modèle d'exigences (Pm_{EMR}), alors que le deuxième illustre la puissance calculée dans le modèle Dymola (Pm_{EMMP}). Le dernier objet met en valeur la puissance sélectionnée dans le modèle COTS (Pm_{EMC}). Trois morphismes se présentent dans la catégorie de la puissance. Un morphisme entre (Pm_{EMR}) et (Pm_{EMMP}) pour illustrer l'acti-

vité multi-physique, autrement dit, le passage du modèle d'exigences vers le modèle multi-physique. Le deuxième morphisme représente l'activité de choix des composants, avec *COTS activity* : $(Pm_{EMMP}) \rightarrow (Pm_{EMC})$. Un troisième morphisme se présente également dans cette catégorie pour illustrer l'activité de vérification entre le modèle d'exigences et le modèle COTS. Ce morphisme est considéré dans le contexte de la théorie des catégories, comme la composition du morphisme *multiphysical activity* avec le morphisme *COTS activity* (la flèche en pointillées dans la figure 4.12). Chaque objet dans cette catégorie, dispose d'un morphisme d'identité pour représenter son évolution interne. Ces morphismes ne seront pas représentés dans toutes les catégories pour éviter l'encombrement.

Prenons un autre exemple pour mieux illustrer notre approche CKIP, celui de la catégorie de coût. Cette catégorie est illustrée par la figure (4.13) et contient deux objets différents. Le premier objet (Ct_{EMR}) met en valeur l'exigence sur le coût définie dans le modèle d'exigences et le deuxième objet (Ct_{EMC}) illustre le coût obtenu dans le modèle COTS. Un morphisme entre ces deux objets est créé pour représenter l'activité de choix des composants.

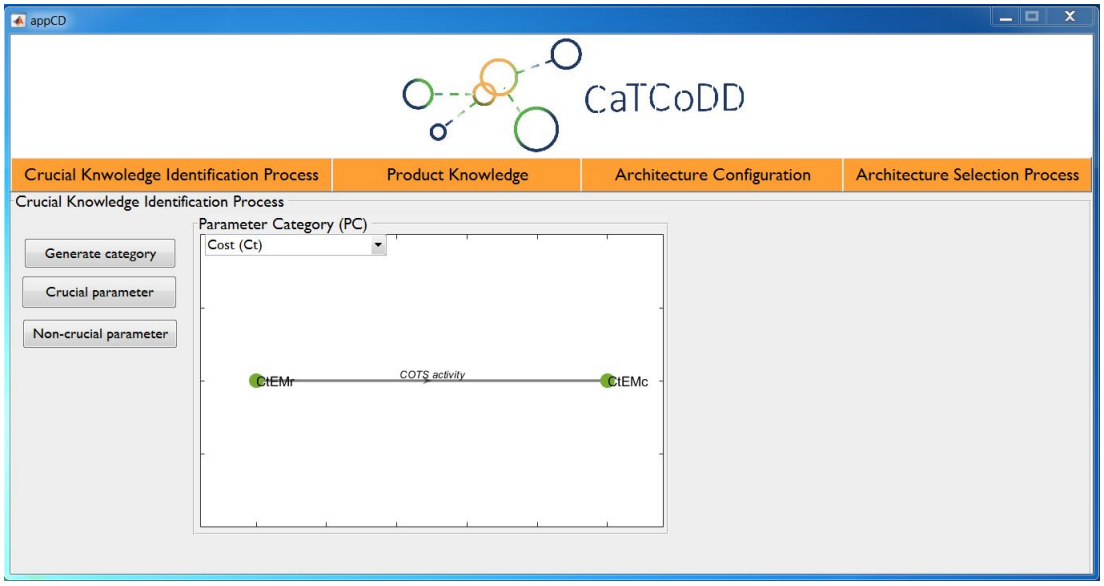


FIGURE 4.13 – Catégorie de coût dans le démonstrateur CaTCoDD

Le troisième exemple illustre la catégorie du rendement du moteur (Eff_{mot}). Comme indiqué dans la figure (4.14), cette catégorie contient un seul objet représentant le rendement du moteur obtenu depuis le modèle COTS (Eff_{motEMC}).

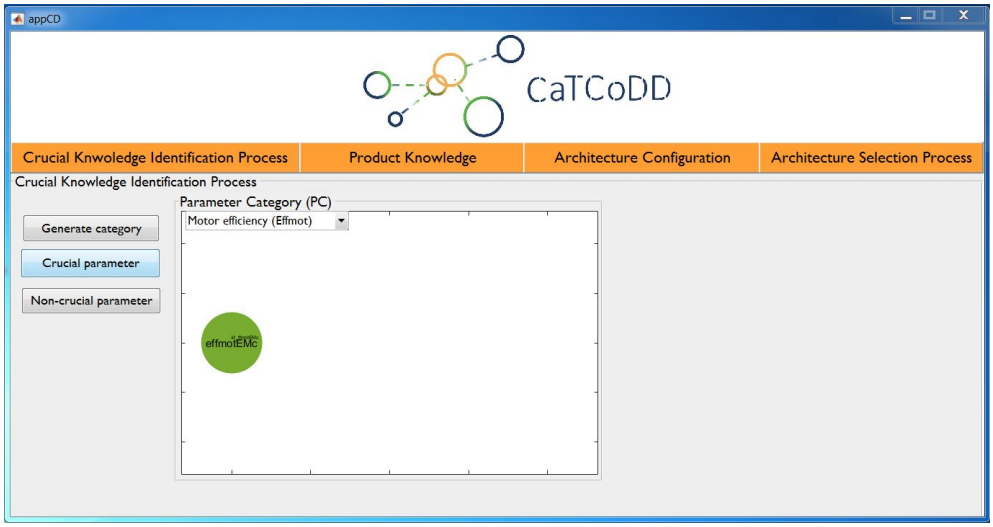


FIGURE 4.14 – Catégorie de rendement du moteur

On procède de la même manière pour l'ensemble des paramètres de l'EMA. Une fois toutes les catégories nécessaires créées, nous pouvons identifier les paramètres cruciaux. Les paramètres représentés avec des catégories discrètes (définie dans la section 1.10.2) ne seront pas considérés cruciaux à la collaboration. Cependant, les paramètres décrits avec des catégories composées de plus que deux objets seront comptés comme des paramètres cruciaux. La classification finale des paramètres de l'EMA dans CaTCoDD est illustrée par la figure (4.15).

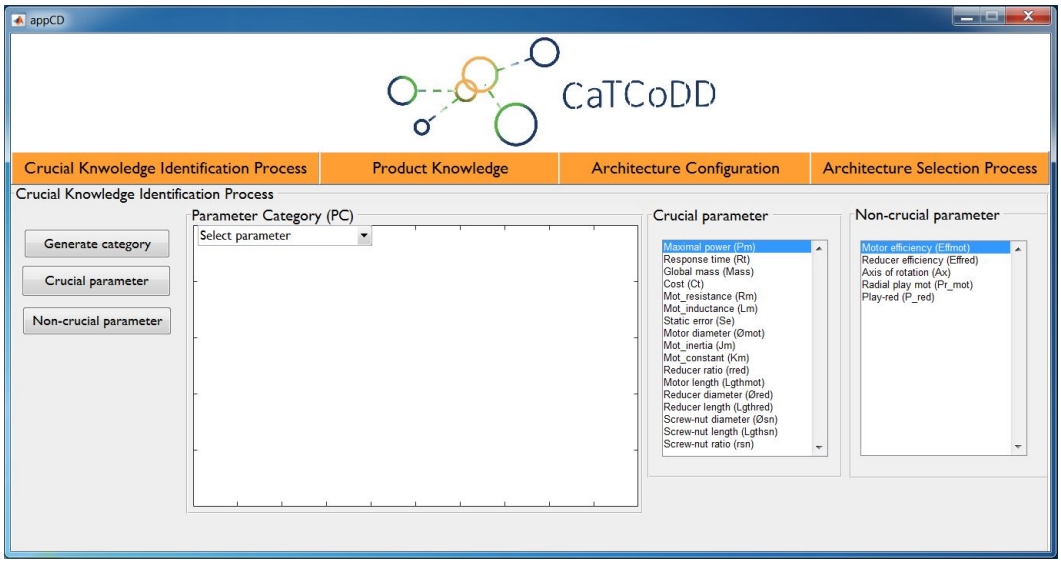


FIGURE 4.15 – Paramètres cruciaux et non cruciaux dans le démonstrateur CaTCoDD

4.3.2 Capitalisation des connaissances cruciales

Une fois les paramètres cruciaux extraits, nous pouvons procéder à leur capitalisation. Pour ce faire, le concept d'entité de base d'information ou *Information Core Entity* (ICE) présenté dans le modèle CDPPK [7] sera utilisé. Ces entités ont pour but de stocker les paramètres nécessaires dans une structure organisée. Notre contribution est de définir les modèles métiers associés pour chaque paramètre dans les ICEs. Cela facilite considérablement la traçabilité des connaissances et permet d'informer le chef de projet sur la source des conflits détectés. Dans la connaissance du produit (*Product Knowledge* (PK)), les ICEs sont définies et les premiers paramètres provenant des exigences sont remplis dans cette étape. Un exemple de création d'une ICE dans CaTCoDD est illustré dans la figure (4.16).

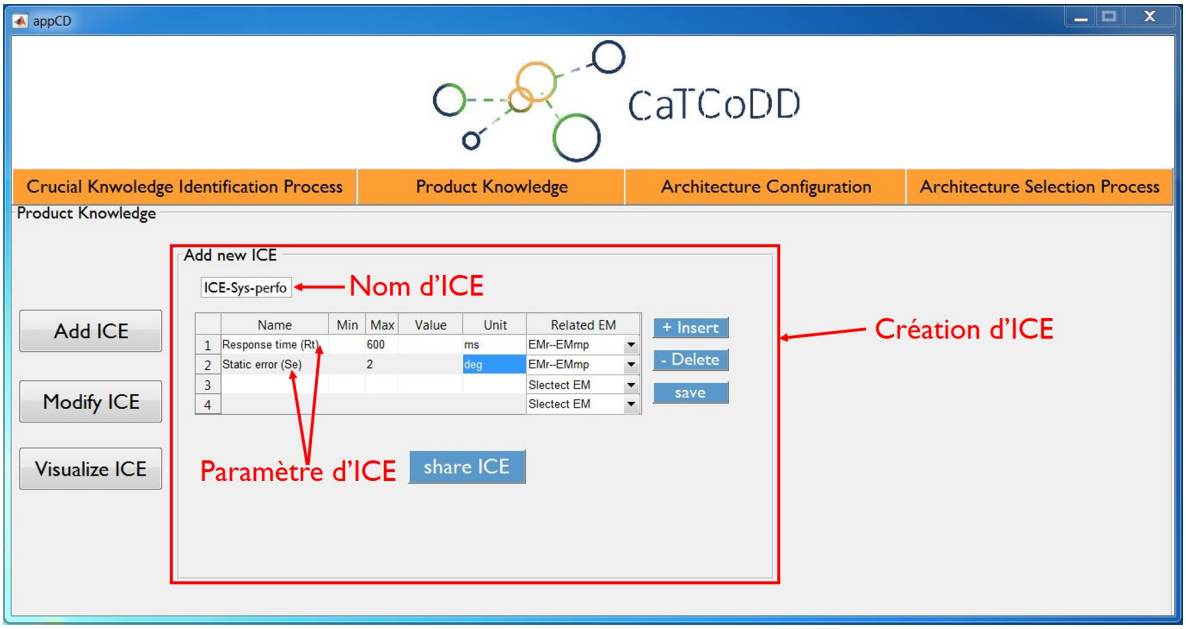


FIGURE 4.16 – Création d'une ICE dans CaTCoDD

Pour la conception de l'EMA, quatre ICEs sont créées pour capitaliser les connaissances cruciales :

— Performances du système : ICE-Sys-perfo
— Caractéristiques du moteur : ICE-Mot-Charact
— Caractéristiques du réducteur ICE-Red-Charact
— Caractéristiques de la vis-écrou ICE-Sn-Charact

Les ICEs créées dans le démonstrateur CaTCoDD sont représentées par la figure (4.17).

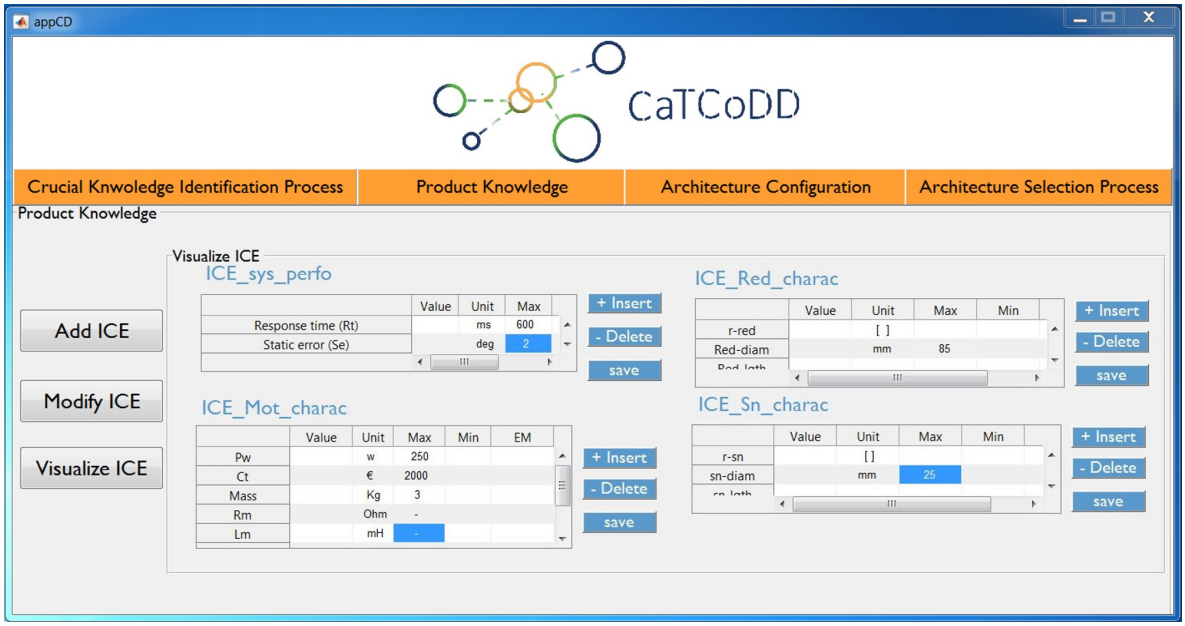


FIGURE 4.17 – Capitalisation des connaissances avec les ICEs

Les ICEs créées peuvent être modifiées et de nouvelles ICEs peuvent également être ajoutées (voir figure 4.17). Par conséquent, le chef de projet partage les ICEs afin qu'elles soient instanciées puis complétées par les ingénieurs disciplinaires dans les étapes suivantes.

4.3.3 Collaboration entre les modèles métiers et les ingénieurs disciplinaires

Après avoir capitalisé les connaissances cruciales à l'aide d'ICEs, la collaboration peut commencer. A ce niveau, deux configurations d'architectures (*Architecture Configuration* AC) sont créées pour illustrer les deux architectures de l'EMA. Pour chaque architecture un ensemble de configurations d'utilisateurs (*User Configuration* UC) est établi.

Quatre UCs correspondantes au quatre modèles métiers décrits au préalable, sont impliqués dans ce processus de collaboration :

— UC-Exigences : permet de définir les exigences qui doivent être respectées afin d'obtenir un système efficace. La comparaison de ces exigences avec les résultats obtenus au cours de la collaboration facilitera le processus de résolution des conflits.
— UC-Dynamic : en utilisant le langage Modelica basé sur des équations différentielles, discrètes et algébriques, cette UC est conçue pour décrire la réponse

dynamique du système [140].
— UC-COTS : dans cette configuration, les composants COTS existants sont sélectionnés conformément aux résultats de la simulation.
— UC-3D : l'intégration 3D de l'ensemble du système est vérifiée tout en tenant compte des exigences ainsi que des modèles métiers précédents.

Chaque ingénieur disciplinaire instancie les paramètres nécessaires pour chaque architecture dans les UCs correspondantes et publie ensuite son travail afin qu'il soit visualisé par tous les collaborateurs. La figure (4.18) illustre les UCs définies pour la conception de l'EMA à entraînement direct. En se basant sur l'idée proposée dans [7], les ICEs sont décomposables ce qui donne les ingénieurs disciplinaires la liberté d'instancier uniquement les paramètres nécessaires à leurs UCs sans besoin d'instancier l'ICE globale. Pour l'architecture à 3 barres, les quatre ICEs créées précédemment sont instanciées dans les différentes UCs. Cependant, l'ICE qui englobe les caractéristiques de la vis-écrou (ICE-Sn-Charact) n'est pas instanciée dans les différentes UCs de l'architecture à entraînement direct vu l'absence de ce composant dans cette architecture.

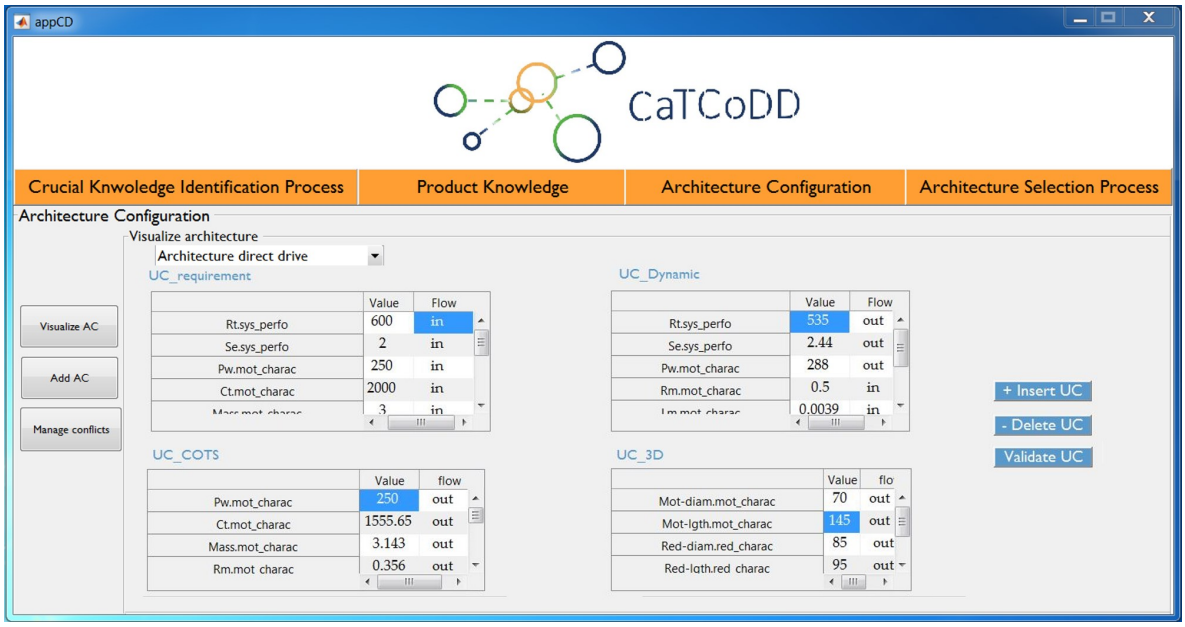


FIGURE 4.18 – Les configurations d'utilisateurs développées dans CaTCoDD pour l'EMA à entraînement direct

Le tableau (4.1) illustre tous les paramètres cruciaux nécessaires au scénario collaboratif de l'EMA avec l'architecture à 3 barres et l'architecture à entraînement direct. Certains paramètres sont définis par le chef de projet en se référant aux exigences présentées précédemment. Néanmoins, les paramètres techniques sont définis par les ingénieurs disciplinaires sur la base des résultats obtenus à partir des

différents modèles métiers.

TABLE 4.1 – Les paramètres cruciaux obtenus à partir des modèles métiers (3B : l'architecture de l'EMA à 3 barres et DD : l'architecture à entraînement direct ou (*Direct Drive*))

Paramètre	**Unité**	**UCreq**	**UC-Dynam**		**UC-COTS**		**UC-3D**	
			Alternatives					
			3B	*DD*	*3B*	*DD*	*3B*	*DD*
ICE-sys-perfo								
Temps de réponse (Rt)	ms	600	530	1760	-	-	-	-
Erreur statique (Se)	deg	2	2.5	15	-	-	-	-
ICE-mot-charact								
Puissance maximale (Pm)	W	250	288	240	250	250	-	-
Coût (Ct)	€	2000	-	-	1555	1155	-	-
Masse (Mass)	Kg	3	-	-	4.143	4.724	-	-
Résistance moteur (Rm)	Ohm	-	0.5	0.3	0.356	0.356	-	-
Inductance moteur (Lm)	mH	-	0.0039	16^{-5}	16.1^{-5}	16.1^{-5}	-	-
Inertie moteur (Jm)	$Kg.m^2$	-	10^{-9}	14^{-5}	13.45^{-5}	13.45^{-5}	-	-
Diamètre moteur (Ømot)	mm	70	-	-	65	65	70	70
Longueur moteur (Lmot)	mm	145	-	-	131.4	131.4	145	145
ICE-red-charact								
Diamètre réducteur (Øred)	mm	85	-	-	81	81	85	85
Longueur réducteur (Lred)	mm	140	-	-	91.9	135.2	95	140
Ratio réducteur (r_{red})	-	-	3.5	14	3.7	14	-	-
ICE-sn-charact								
Diamètre vis-écrou (Øsn)	mm	25	-	-	22	-	25	-
Longueur vis-écrou (Lsn)	mm	60	-	-	58.4	-	60	-
Ratio vis-écrou (r_{sn})	-	-	330	-	333	-	-	-

4.3.4 Gestion des conflits

Dans cette étape, les conflits survenus au cours de la collaboration sont vérifiés. Le processus de résolution des conflits est basé sur le formalisme de la théorie des catégories tel que détaillé précédemment. À ce niveau, les catégories des paramètres (PC), établies lors du processus d'identification des connaissances cruciales, sont mises à jour et des valeurs peuvent être attribuées à chaque paramètre. Un exemple de l'instance de la catégorie de l'erreur statique est illustré par la figure (4.19).

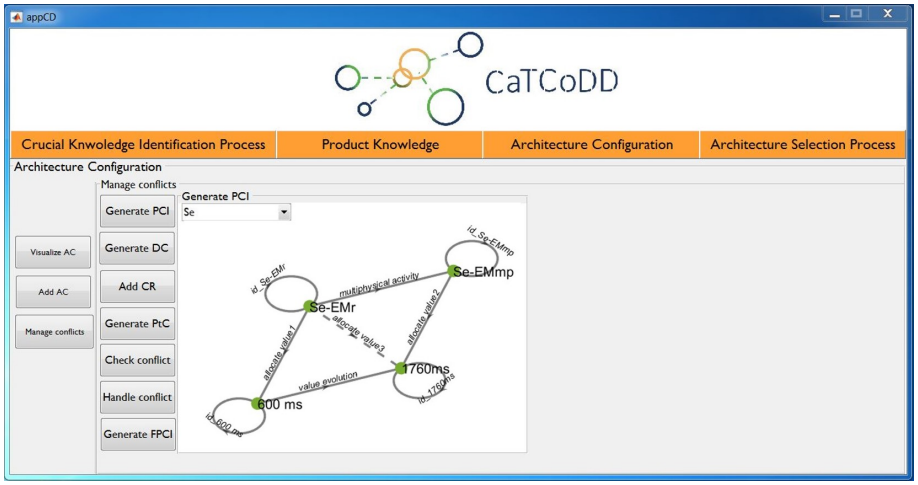


FIGURE 4.19 – PCI de l'erreur statique pour l'architecture à entraînement direct dans le démonstrateur CaTCoDD

Une fois la PCI générée pour tous les paramètres cruciaux, une catégorie de dépendance (DC) est établie. La DC vise à mettre en évidence le degré de dépendance entre les paramètres. Chaque objet de cette DC représente un paramètre et chaque morphisme fait référence à un coefficient de dépendance entre les paramètres. Par conséquent, les décisions pour gérer les conflits dépendront de ces coefficients, comme sera expliqué dans la suite. Les DCs du système EMA à 3 barres et à entraînement direct sont illustrées par la figure (4.20) et la figure (4.21) respectivement.

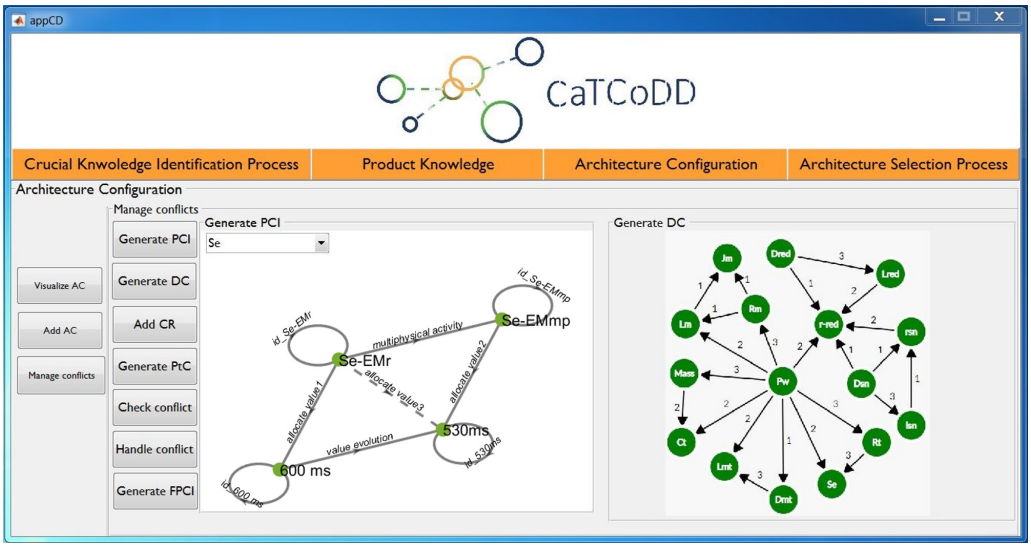


FIGURE 4.20 – La catégorie de dépendance pour l'architecture à 3 barres

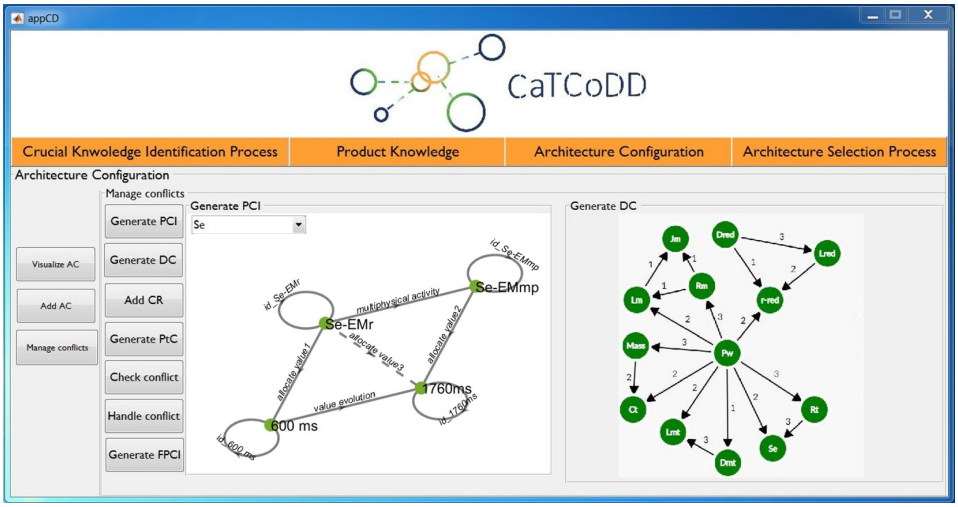


FIGURE 4.21 – La catégorie de dépendance pour l'architecture à entraînement direct

La différence entre ces deux catégories réside dans le fait que dans la DC de l'EMA à entraînement direct, les paramètres de la vis-écrou (la longueur (L_{sn}), le diamètre (Øsn) et le ratio (r_{sn})) sont supprimés.

Une fois les catégories de dépendances de chaque architecture établies, un ensemble de règles de cohérences (*Consistency Rules* CRs), faisant référence à des contraintes qui doivent être respectées par les modèles métiers impliqués dans la conception, est défini. Ces règles permettent de décrire la relation entre les différentes instances d'un même paramètre. Un extrait de règles de cohérences établies dans CaTCoDD est illustré par la figure (4.22).

Ces règles seront par la suite traduites en catégorie de patterns. L'objectif de ce pattern est de représenter les conflits qui peuvent survenir. La catégorie de pattern (*Pattern Category* (PtC)) est composée d'un ensemble d'objets et de morphismes comme expliqué précédemment. Deux types d'objets existent dans ce cas, les objets constants qui représentent les noms des paramètres et les objets variables qui font référence aux valeurs des paramètres. Les objets sont reliés entre eux par des morphismes qui formulent le non-respect d'une règle de cohérence. Un exemple d'une PtC est représentée dans notre démonstrateur à travers la figure (4.23).

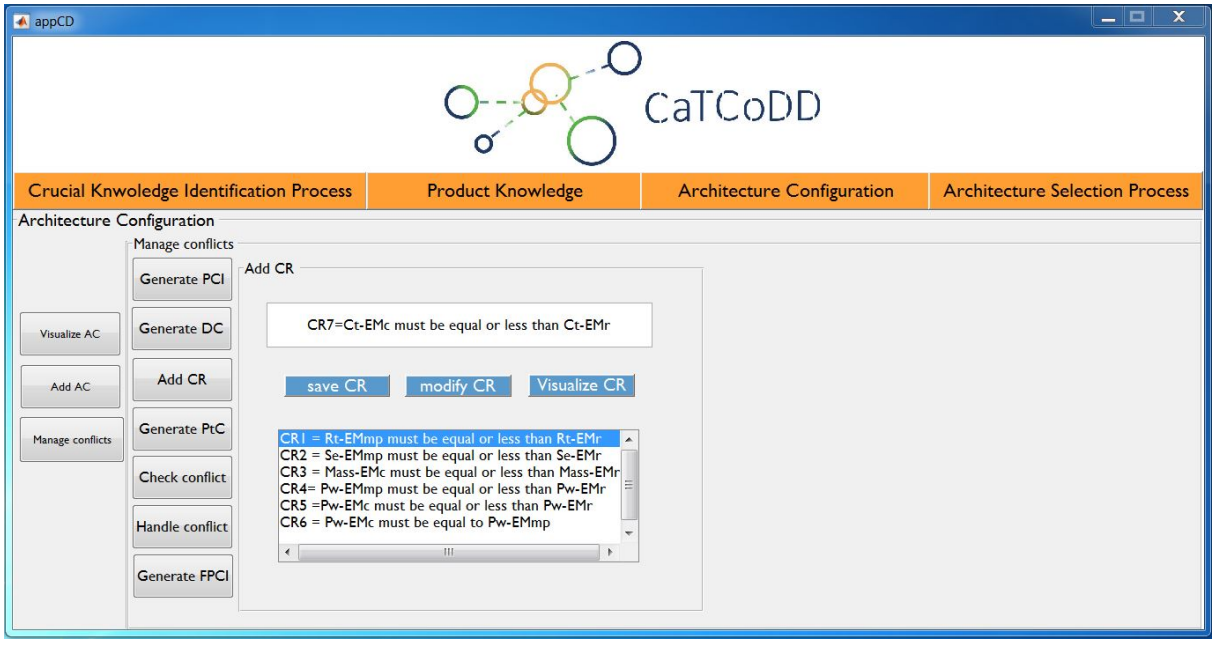


FIGURE 4.22 – Définition des règles de cohérences dans CaTCoDD

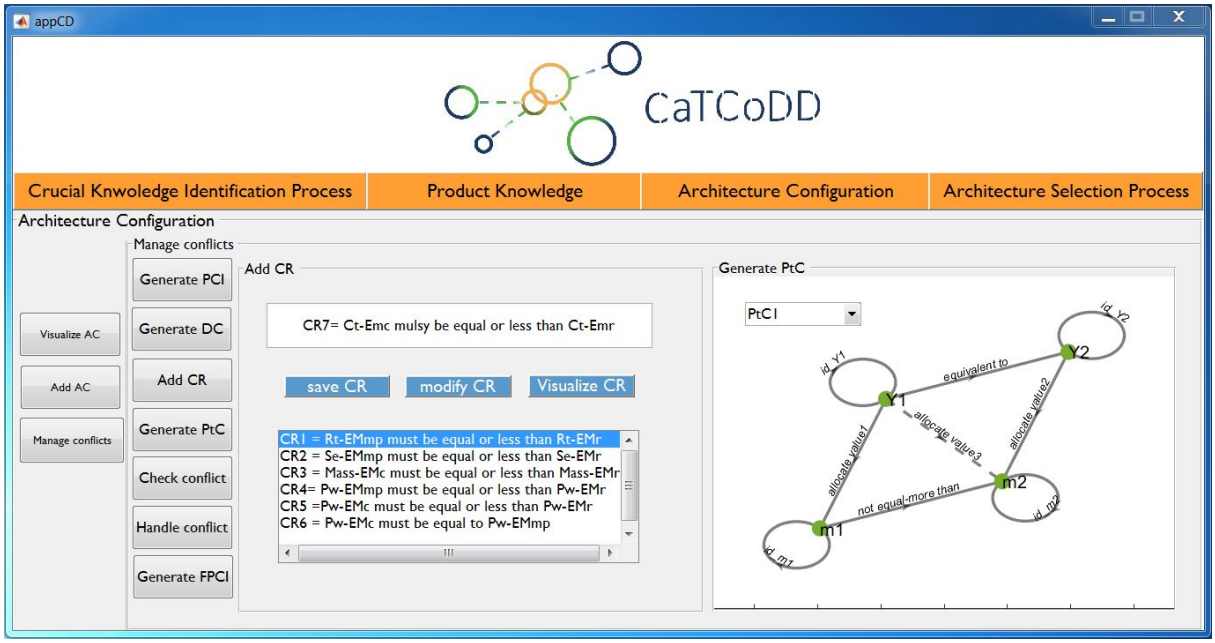


FIGURE 4.23 – Catégorie de pattern (PtC) dans CaTCoDD

Dans la conception de l'EMA, on se retrouve devant deux catégories de patterns. La première représente un conflit entre deux modèles métiers comme le montre la

figure (4.23) et la deuxième illustre un conflit qui peut apparaître entre trois modèles métiers différents (voir figure 4.24).

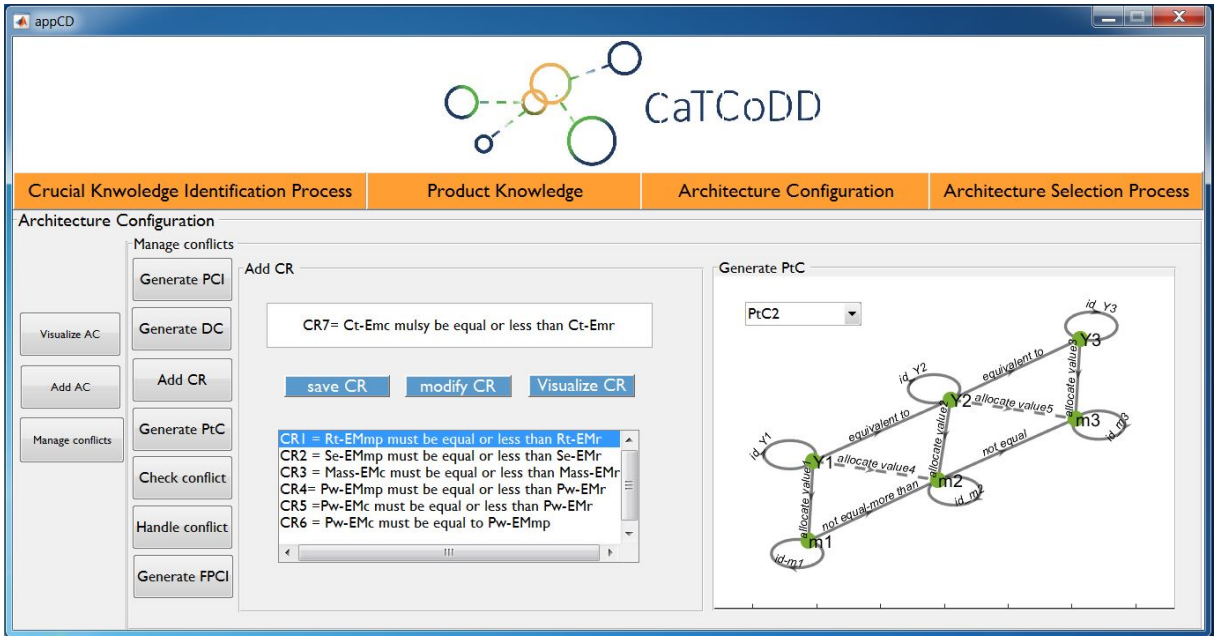


FIGURE 4.24 – Catégorie de pattern pour un conflit entre trois modèles métiers

Après avoir créé les catégories de pattern nécessaires au processus de résolution des conflits, le chef de projet peut vérifier les conflits existants afin de les traiter et obtenir un système cohérent. La vérification des conflits est basée sur le concept formel de la théorie des catégories, appelé "foncteur" comme nous l'avons déjà souligné précédemment (voir section 3.8.5). Par conséquent, un mapping entre les catégories des paramètres et des patterns est nécessaire pour détecter les conflits. Ce mapping nous permet de visualiser les conflits survenus ainsi que les modèles métiers concernés. Les conflits apparus après la collaboration entre les modèles métiers dans l'architecture à 3 barres sont représentés par la figure (4.25).

De plus, pour l'architecture de l'EMA à entraînement direct des conflits sont également détectés en se basant sur le mapping entre les différentes catégories des paramètres et les catégories des patterns. Ces conflits sont illustrés par la figure (4.26).

À ce niveau final, les conflits détectés entre les différentes valeurs des paramètres doivent être soigneusement gérés. Un ensemble d'actions de résolution est appliqué à ces conflits tout en tenant compte les dépendances entre les paramètres. Par conséquent, chaque modification d'un paramètre implique une modification du paramètre lié. Le flux de dépendance des paramètres concernés par les conflits est illustré par la figure (4.27).

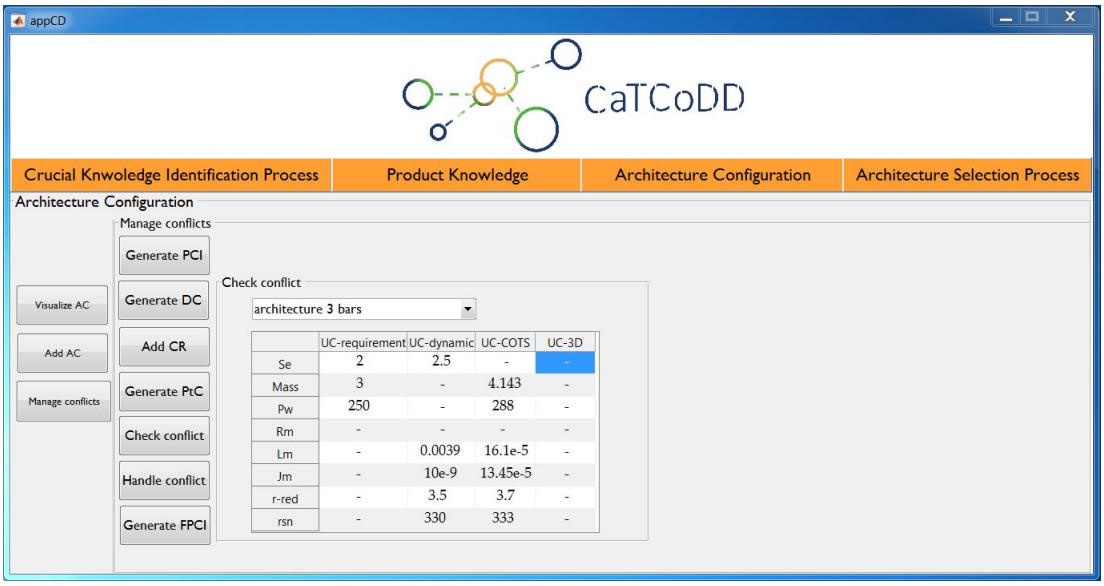


FIGURE 4.25 – Conflits détectés dans l'EMA à 3 barres

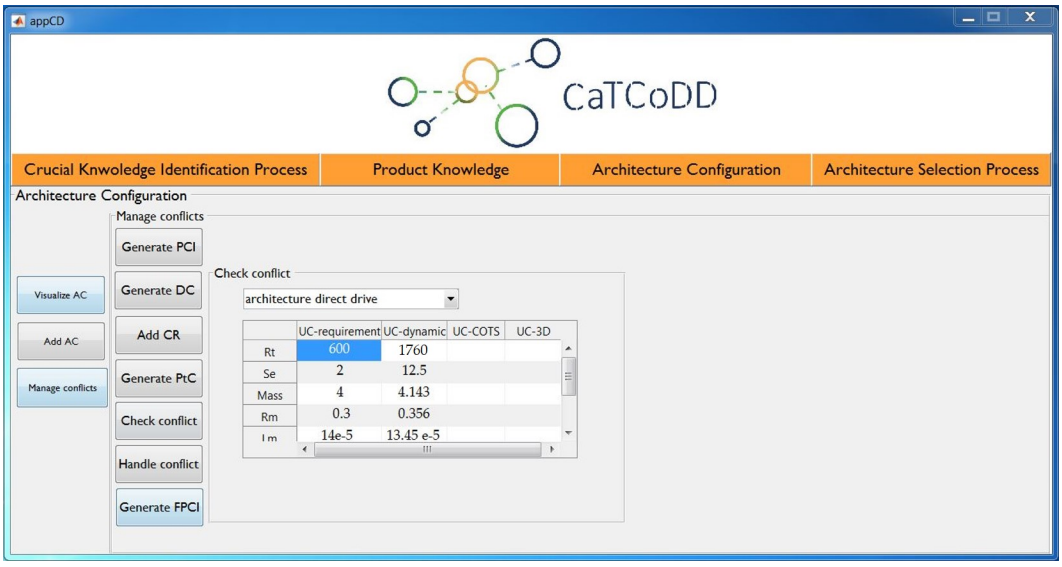


FIGURE 4.26 – Conflits détectés dans l'EMA à entraînement direct

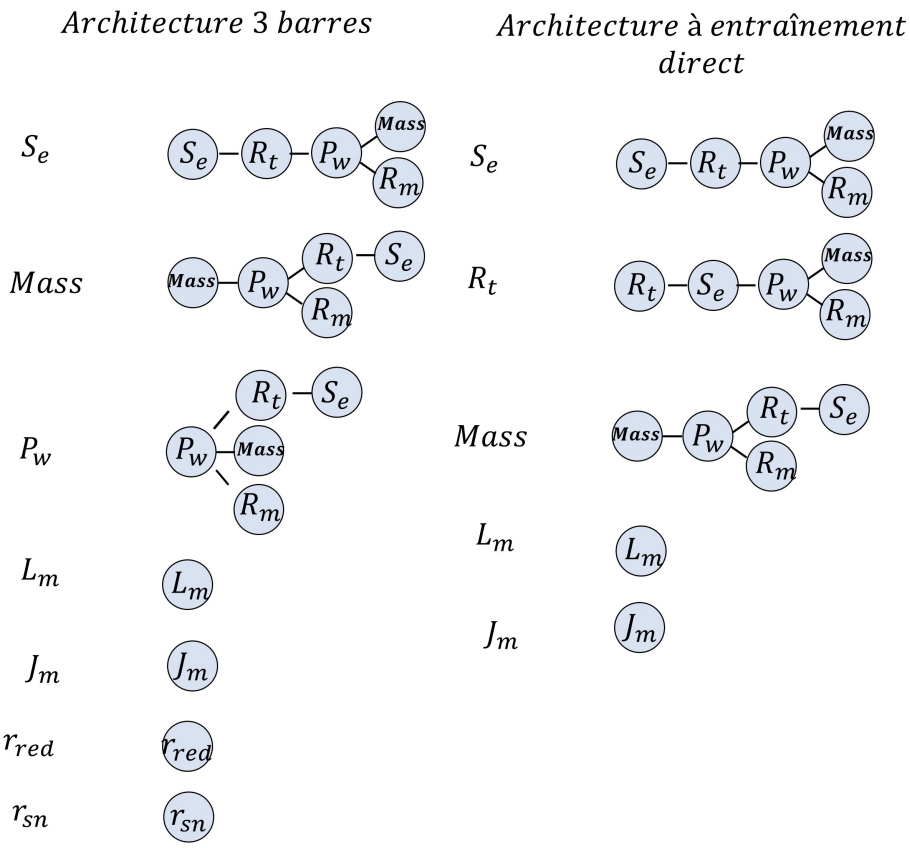


FIGURE 4.27 – Dépendance entre les paramètres concernés par les conflits

Les conflits dans les valeurs de la résistance (R_m), l'inductance (L_m) et l'inertie (J_m) du moteur durant la conception de l'EMA à entraînement direct ont été ignorés. Cependant, une tolérance a été proposée à travers une libération de l'exigence sur la masse. Pour cela, l'exigence de la masse passe de 3 Kg vers 5 Kg. Par ailleurs, pour résoudre les conflits dans l'erreur statique (S_e) et le temps de réponse (R_t) des modifications dans les modèles multi-physiques doivent avoir lieu. On commence cette modification par l'action sur les paramètres du contrôleur PID afin d'améliorer les performances de la réponse du moteur. Cependant, nous avons constaté que ces modifications n'ont pas pu agir sur le temps de réponse et l'erreur statique. De ce fait, une modification dans les paramètres du moteur et du réducteur dans le modèle multi-physique sera établie ce qui impliquera également une modification dans les paramètres du moteur sélectionnés dans le modèle COTS. Ces modifications sont illustrées par la figure (4.28).

D'après ces modifications, on constate que l'erreur statique a été réduite. Cependant, les valeurs du temps de réponse et de la masse sont plus élevées après la

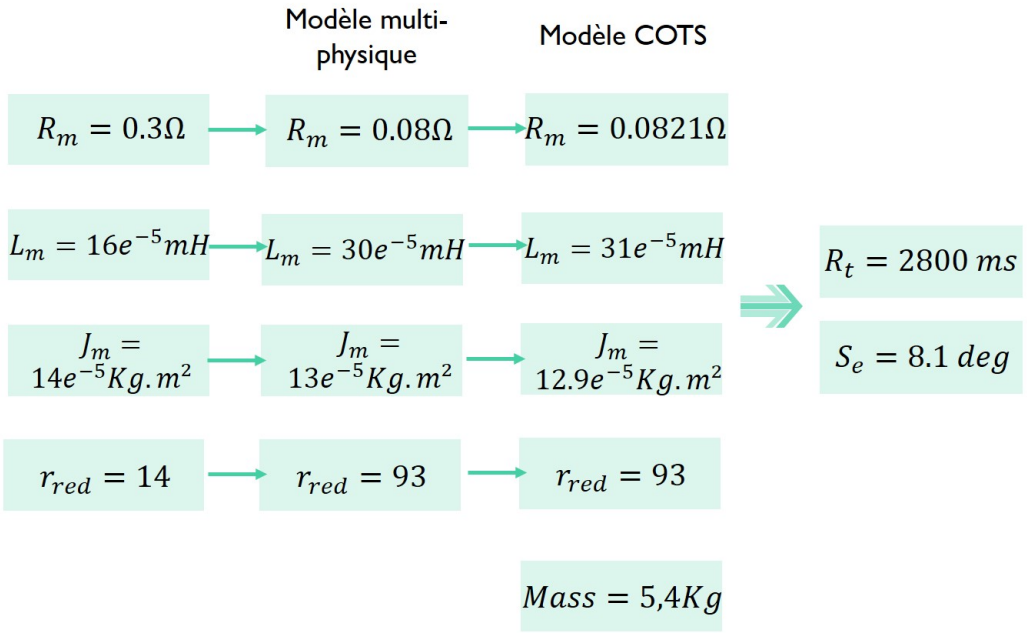


FIGURE 4.28 – Modifications effectuées pour résoudre les conflits dans l'architecture à entraînement direct de l'EMA

modification. De ce fait, les exigences ne sont pas toujours respectées. En ce sens, nous décidons de garder la première solution et ignorer les conflits dans l'erreur statique et le temps de réponse.

Pour la deuxième architecture de l'EMA à trois barres, un ensemble d'actions de résolution a été appliqué afin de gérer les conflits. Les conflits de la masse, l'inductance et l'inertie du moteur nécessite une action de modification afin de gérer les conflits. Pour cela, nous agissons sur le modèle multi-physique afin de modifier la valeur de ces paramètres. Les paramètres qui subissent des modifications sont représentés dans la figure (4.29). Nous pouvons remarquer que suite à la dépendance entre la puissance et l'erreur statique ainsi que le temps de réponse, la valeur de l'erreur statique est modifiée où un passage de 2.5 deg vers 1.96 deg a eu lieu. De ce fait, le conflit entre les deux valeurs de l'erreur statique a été résolu. Nous pouvons également constater une légère amélioration dans le temps de réponse comme illustré sur la figure (4.30).

Par ailleurs, le conflit dans la masse est toléré en appliquant une libération d'exigence sur ce paramètre. Cette libération a eu lieu durant la résolution des conflits dans l'architecture à entraînement direct pour avoir une nouvelle valeur égale à 5 Kg. Avec cette tolérance, l'exigence sur la masse est respectée et le conflit est résolu. Finalement, les incohérences dans les valeurs des ratios de réducteur et de la vis-écrou sont ignorées.

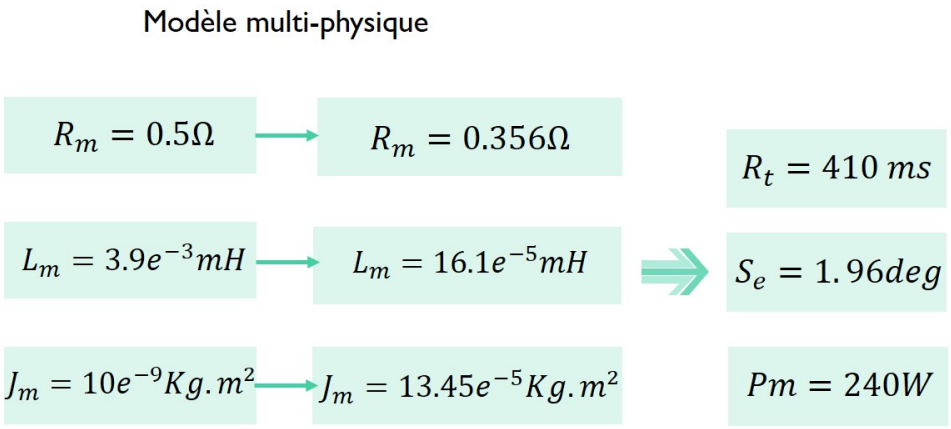


FIGURE 4.29 – Modifications effectuées pour résoudre les conflits dans l'architecture à trois barres

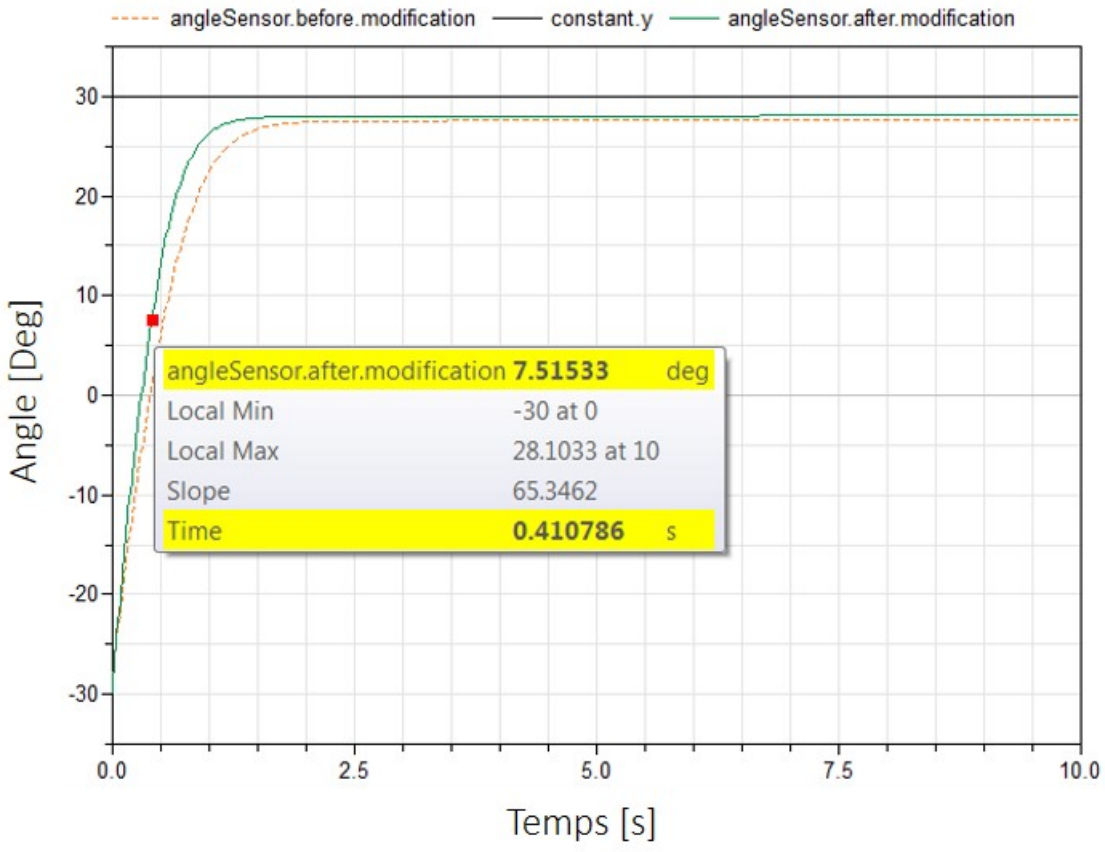


FIGURE 4.30 – Temps de réponse avant et après modification dans l'architecture à trois barres

Suite à ces modifications dans les deux architectures de l'EMA, les nouvelles valeurs seront sauvegardées dans les instances des catégories des paramètres finales (*Final Parameter Category Instance* FPCI) comme nous l'avons détaillé dans le chapitre précédent. A titre d'exemple, la FPCI de l'erreur statique dans l'EMA à trois barres est représentée par la figure (4.31). En ce qui suit, nous allons utiliser ces valeurs pour comparer les deux architectures et choisir l'architecture qui répond le mieux aux besoins.

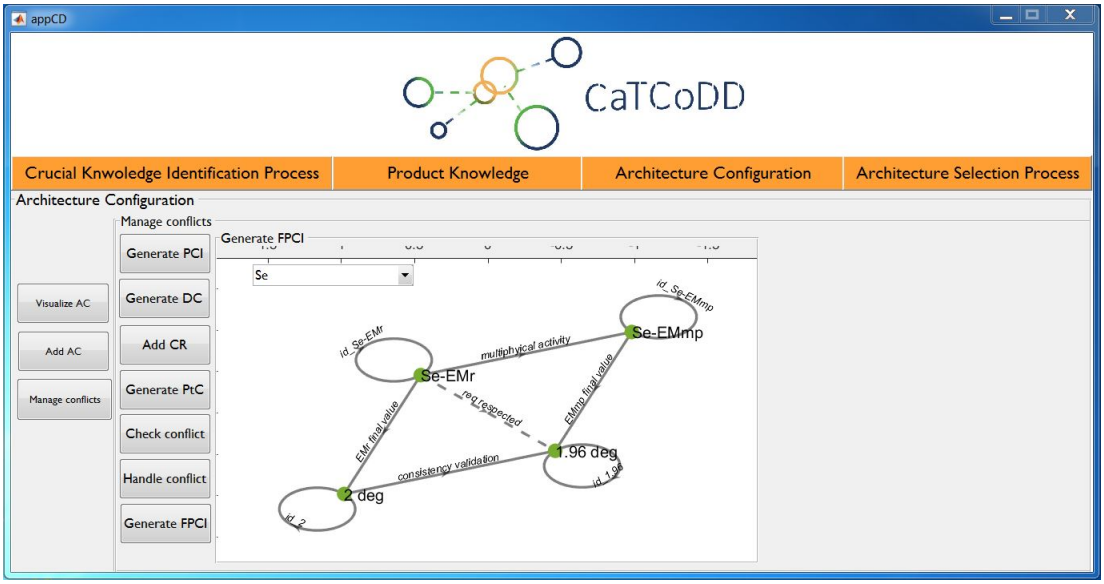


FIGURE 4.31 – FPCI de l'erreur statique dans l'EMA à trois barres

4.4 Étape 3 : Prise de décision et choix d'architectures

Dans cette dernière étape, une comparaison entre les deux architectures de l'EMA sera effectuée afin de choisir l'architecture qui satisfait au mieux les critères. Cette étape se résume en quatre sous-étapes. La première sous-étape consiste à identifier les critères de comparaison. Durant la deuxième sous-étape, le recueil de données relatives à chaque architecture a lieu et le calcul des scores s'effectue au cours de la troisième sous-étape. Finalement, en se basant sur les scores calculés, une décision peut être prise. Les différentes sous-étapes seront appliquées à l'EMA dans les parties suivantes.

4.4.1 Identification des critères de comparaison

Dans la conception de l'EMA, trois critères de comparaison sont identifiés : le temps de réponse, l'erreur statique et le coût. En se basant sur ces critères, la comparaison entre les deux architectures de l'EMA sera effectuée. Ces critères sont introduits dans le démonstrateur comme le montre la figure (4.32).

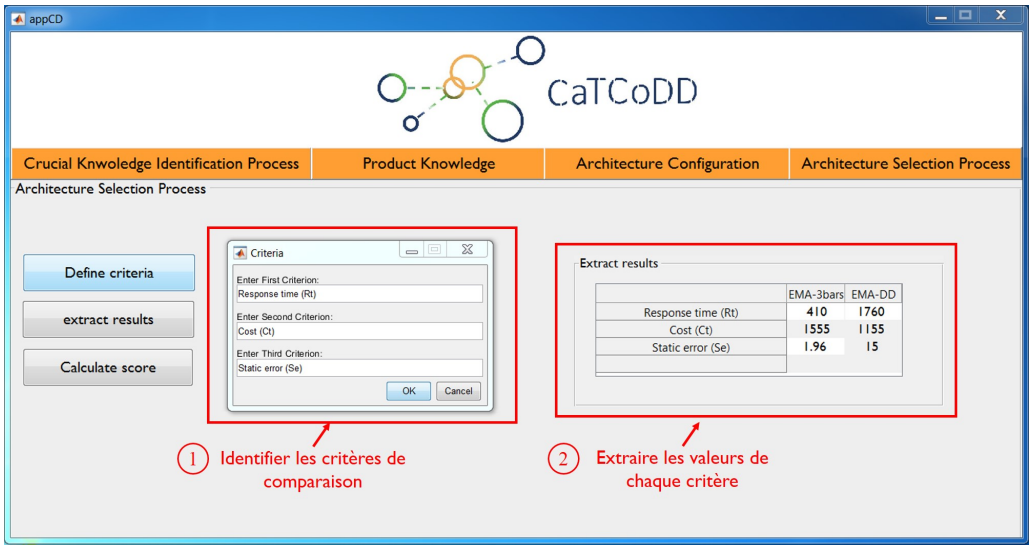


FIGURE 4.32 – Identification des critères de comparaison et les valeurs pour les deux architectures de l'EMA

4.4.2 Recueil de données relatives à chaque architecture

A ce niveau, la valeur de chaque critère de comparaison est identifiée. La figure (4.32) illustre les différentes valeurs de l'erreur statique, le temps de réponse et le coût pour l'EMA à entraînement direct et à trois barres.

4.4.3 Calcul de score pour chaque alternative en se basant sur la méthode AHP

En suivant les étapes détaillées dans la section (3.13), nous calculons le score de chaque architecture de l'EMA. Nous commençons par créer la matrice de jugement et vérifier la cohérence de ces jugements comme nous pouvons le voir dans la figure (4.33).

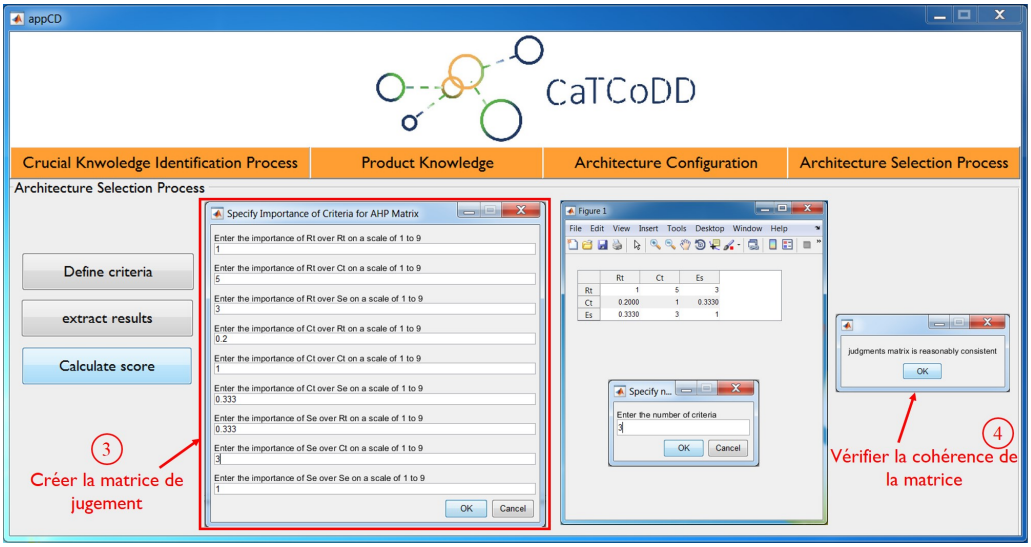


FIGURE 4.33 – Matrice de jugement créée dans le démonstrateur CaTCoDD et vérification de cohérence

En ce qui suit, nous passons au calcul des priorités relatives des architectures par rapport au temps de réponse, coût et erreur statique. Pour cela, une comparaison par paire des architectures par rapport à ces trois critères est nécessaire comme représenté sur la figure (4.34). Finalement, les scores d'architectures sont calculés comme illustré par la figure (4.35).

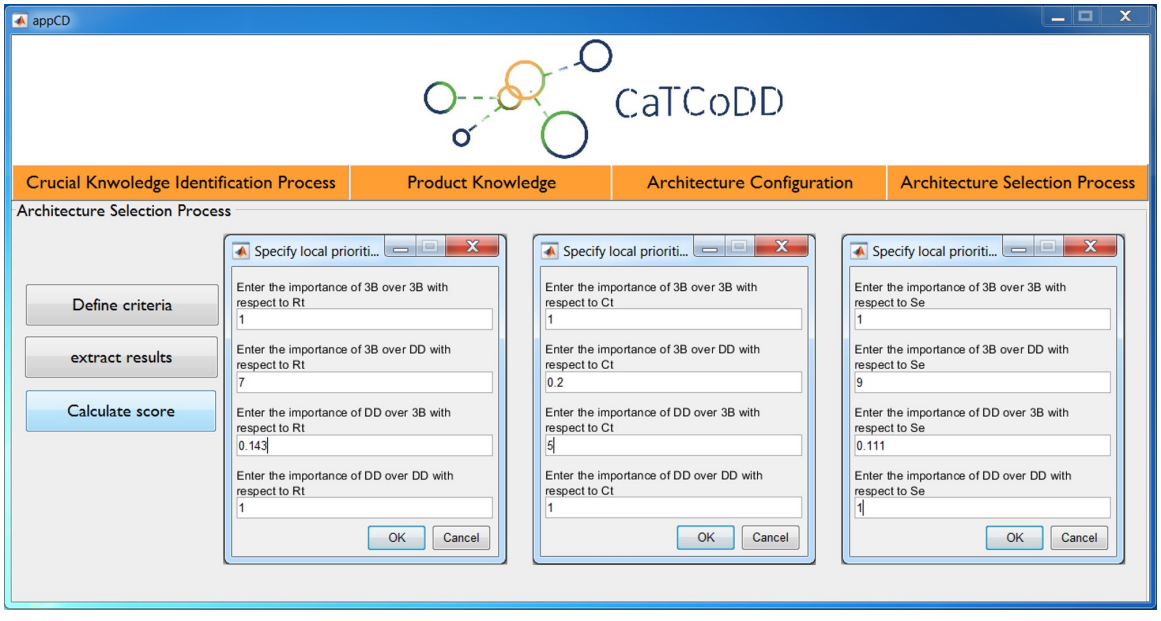


FIGURE 4.34 – Priorités obtenues par rapport aux critères de comparaison

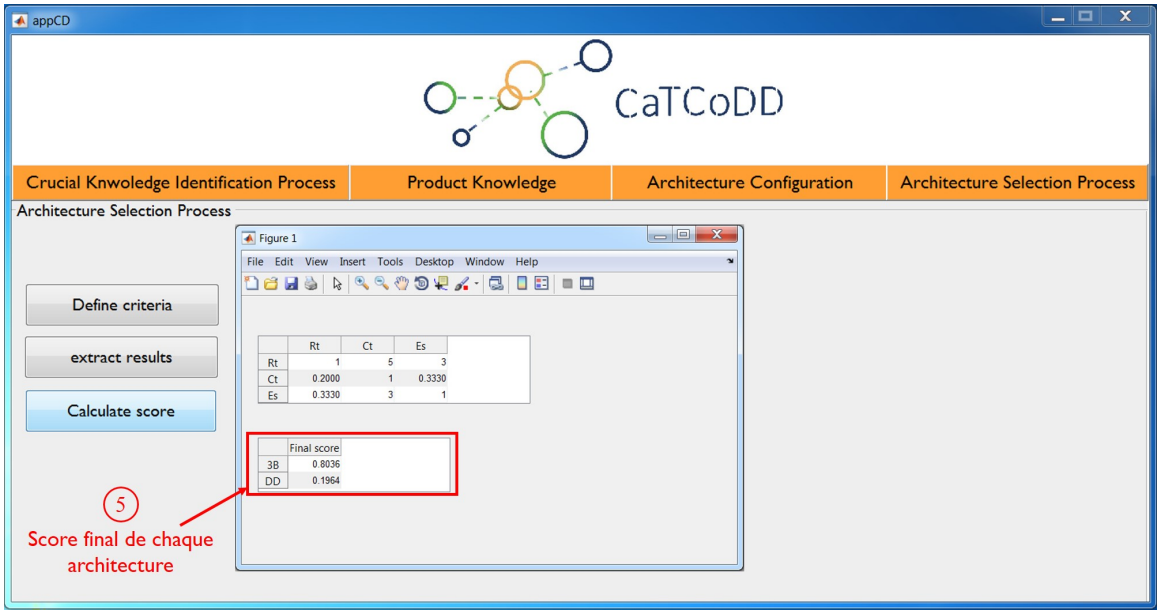


FIGURE 4.35 – Score final de l'architecture à trois barres et à entraînement direct

4.4.4 Choix d'architecture

Finalement, en se basant sur les scores obtenus pour chaque architecture de l'EMA, nous pouvons constater que l'EMA à trois barres possède un score plus important que la deuxième architecture à entraînement direct. De ce fait, l'architecture à trois barres répond le mieux aux critères et elle est considérée comme l'architecture la plus appropriée.

4.5 Comparaison entre CaTCoDD et les outils existants

Afin d'assurer une collaboration efficace entre le chef de projet, l'ingénieur système et les ingénieurs disciplinaires dans le contexte de la conception mécatronique, différentes méthodes ont été proposées comme nous l'avons souligné tout au long de ce manuscrit. D'une part, dans leurs travaux de recherche, Dave et Koskela [142] ont affirmé que les outils de gestion de l'information, tels que les courriels, l'intranet, l'extranet et les systèmes de gestion des documents, peuvent avoir un impact négatif sur les capacités de gestion des connaissances de l'organisation. Cela est dû au fait que ces outils provoquent une surcharge d'informations en raison d'un échange d'informations non organisé. De ce fait, ces moyens ne sont plus suffisants pour supporter la collaboration entre des modèles hétérogènes car ils conduisent à

des itérations inutiles et à un manque de traçabilité [7].

D'autre part, la méthode KCMethod dévéloppée par Badin et al. [10] a été implémentée sous forme d'un outil de collaboration appelé KARREN (*Knowledge Acquisition and Reuse for Robust ENgineering*). Cet outil a été mis en oeuvre par l'entreprise *Digital Product Simulation* (DPS) afin d'assurer la capitalisation et la réutilisation des connaissances de granulartié fine. KARREN a fait l'objet de quelques travaux de recherche dans le contexte de la conception collaborative des systèmes mécatroniques [139, 58]. Dans les travaux effectués par Mhenni et al. [139], KARREN a été utilisé pour permettre le partage de paramètres entre les équipes de conception et pour effectuer des compromis afin de répondre aux exigences de performances et d'intégration dans le cadre d'une expérimentation du scénario collaboratif de l'actionneur électromécanique d'aileron (Voir figure 4.36). Cet outil a été également utilisé pour illustrer la conception collaborative d'un boîtier papillon (voir figure 4.37) [58].

Reducteur .1		V2		PublicDomain	
Parameters					
Puissance_reducteur			Power		W
Couple_reducteur			Moment		N.m
Vitesse_reducteur			Angular speed		rad/s
Inertie_reducteur			Moment of inertia		kg.m2
Rendement_reducteur			Efficiency		%
Masse_reducteur			Mass		kg
Charge_axiale			Force		N
Charge_radiale			Force		N
Diametre_reducteur			Length		mm
Longueur_moteur			Length		mm
Reference_Catalogue			String		char
Rapport_reduction			Without dimensio		300 Real

FIGURE 4.36 – Utilisation de KARREN pour un scénario de collaboration durant la conception de l'EMA

En se référant à ces travaux de recherche, nous pouvons constater que KARREN est utile pour soutenir le travail collaboratif car il permet de raccourcir les itérations de conception et de faire converger les compromis sur les paramètres des composants vers des solutions acceptables au regard des exigences données. Cependant, cet outil a montré ses limites en termes d'identification des connaissances cruciales. Des échanges entre les participants dans le projet de conception sont effectués afin de se mettre d'accord sur les connaissances qui nécessitent la capitalisation. Cette tâche est chronophage et difficile à effectuer dans un environnement de conception

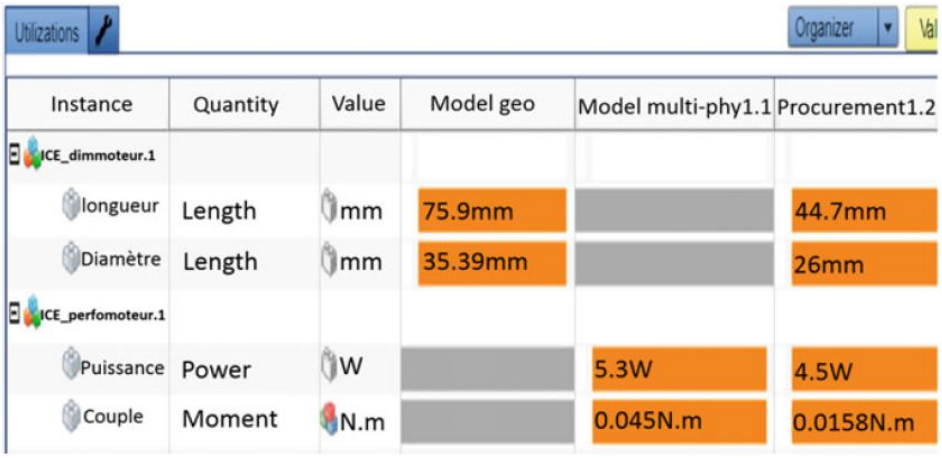


FIGURE 4.37 – Utilisation de KARREN pour un scénario de collaboration durant la conception du boîtier papillon [58]

complexe. De plus, les conflits détectés sont résolus manuellement par le chef de projet sans ayant recours à des actions de résolution explicites ce qui peut influencer la fiabilité de ces décisions. Par ailleurs, l'outil KARREN permet de supporter la conception d'une architecture d'un système mécatronique sans proposer des méthodes pour comparer différentes architectures et aider le chef de projet dans son choix.

Par conséquent, CaTCoDD présente plus de fonctionnalités par rapport à KARREN notamment en termes d'identification formelle des connaissances cruciales en se basant sur la théorie des catégories, de localisation et gestion des conflits en ayant recours aux patterns et règles de cohérences ainsi que d'aide à la décision et choix d'architectures. Ces fonctionnalités permettent de faciliter la collaboration entre différentes parties prenantes d'un projet de conception, de construire un réseau de collaboration efficace et de guider le chef de projet, l'ingénieur système et les ingénieurs disciplinaires depuis la tâche d'identification des exigences jusqu'à l'étape de choix de l'architecture qui répond au mieux aux exigences.

4.6 Conclusion

Dans ce chapitre, nous avons présenté l'expérimentation de notre méthodologie proposée pour la conception collaborative et l'aide à la décision, mettant en scène le démonstrateur CaTCoDD pour la conception d'un actionneur électromécanique d'aileron (EMA). Nous avons utilisé différents modèles disciplinaires et deux architectures afin de réaliser le scénario de collaboration dans CaTCoDD.

Dans un premier temps, nous avons partis de la spécification des exigences et

l'identification d'architectures à comparer. Ensuite, quatre modèles métiers ont été établis pour la conception de l'EMA. Un flux bien précis a été suivi durant la création de ces modèles. Dans un second temps, nous avons passé à la phase de collaboration où on a commencé par l'extraction formelle des connaissances cruciales en se basant sur la théorie des catégories. Ces connaissances ont été capitalisées sous forme d'ICEs. Ensuite, la collaboration et l'échange entre les différents participants au projet de conception a eu lieu dans l'espace de collaboration du démonstrateur CaTCoDD. Deux architectures ont été créées, à savoir l'architecture à trois barres et à entraînement direct. Cette étape a été suivie de la localisation et la gestion des conflits en se basant sur la théorie des catégories. Enfin, la mise en place de la méthode AHP dans le démonstrateur CaTCoDD a été effectuée afin de comparer les deux architectures de l'EMA. Cette méthode nous a permis de conclure que l'architecture à trois barres répond le mieux aux exigences.

Suite aux retours recueillis tout au long de ce manuscrit, nous allons dresser, en ce qui suit, un bilan ainsi qu'une critique de notre méthodologie CaTCoDD et son démonstrateur associé.

Conclusion et perspectives

Conclusion générale

La conception collaborative apparaît comme un aspect fondamental dans le processus de conception mécatronique. Cette collaboration nécessite un environnement commun entre les différents ingénieurs participant au projet de conception afin d'échanger leurs connaissances et obtenir un système cohérent. Cet échange concerne de nombreuses connaissances qui peuvent être considérées cruciales pour des disciplines et inutiles pour d'autres. Par ailleurs, durant cette collaboration des incohérences peuvent se présenter vu la pluridisciplinarité des métiers impliqués à la conception. En plus, le système mécatronique peut avoir plusieurs solutions qui nécessitent l'évaluation afin de choisir celle qui répond le mieux aux exigences.

C'est dans ce cadre que nous avons apporté des réponses à ces quatre questions de recherche :

— Comment faire collaborer efficacement et d'une manière cohérente les différentes disciplines impliquées dans le projet de conception mécatronique ?
— Quelles sont les connaissances à échanger entre les différents modèles métiers ?
— Commet détecter et gérer les conflits entre les différents modèles métiers ?
— Comment choisir la solution qui satisfait au mieux les exigences ?

A la lumière de ces questions, nous avons obtenu les résultats suivants comme illustré par la figure (I) :

— Une méthodologie générale de conception collaborative et aide à la décision pour les systèmes mécatroniques, que nous appelons *Category Theory-based Collaborative Design and Decision-making method* (CaTCoDD). Cette méthodologie est basée sur un ensemble de concepts permettant la gestion des connaissances de granularité fine. En se basant sur les concepts fondamentaux de notre méthodologie CaTCoDD, une formalisation sous forme d'un métamodèle a été également effectuée. Enfin, un effort particulier a été consacré au développement du démonstrateur CaTCoDD afin de mieux illustrer les différentes étapes de la méthodologie proposée et valider ainsi ses aspects fondamentaux en se basant sur une implémentation réelle.
— Un cadre formel, baptisé *Crucial Knowledge Identification Process* (CKIP),

pour l'identification formelle des connaissances cruciales en se basant sur la théorie des catégories. Ce cadre fournit une représentation formelle et unifiée des connaissances impliquées dans la collaboration. Cette représentation a donné une signification formelle aux dépendances entre les différentes parties prenantes. Cette formalisation des connaissances permet d'éviter la nécessité d'un échange direct entre les parties prenantes, qui peut être difficile à organiser particulièrement dans les projets de conception mécatronique.

— Une approche de localisation et gestion des conflits en se basant sur les différents concepts de la théorie des catégories, que nous désignons par *Conflict Resolution Process* (CRP). Le recours aux règles de cohérences et patterns dans cette approche a permis également une localisation formelle et simple des conflits. L'utilisation de ces concepts rend le processus de résolution des conflits plus flexible puisque les règles peuvent être supprimées ou ajoutées sans réviser l'ensemble du mécanisme de gestion des conflits.

— Une intégration de la méthode d'ADMC AHP dans notre méthodologie CaT-CoDD afin de comparer différentes architectures d'un système mécatronique et aider le chef de projet dans son choix. Nous appelons cette approche de choix d'architectures *Architecture Selection Process* (ASP).

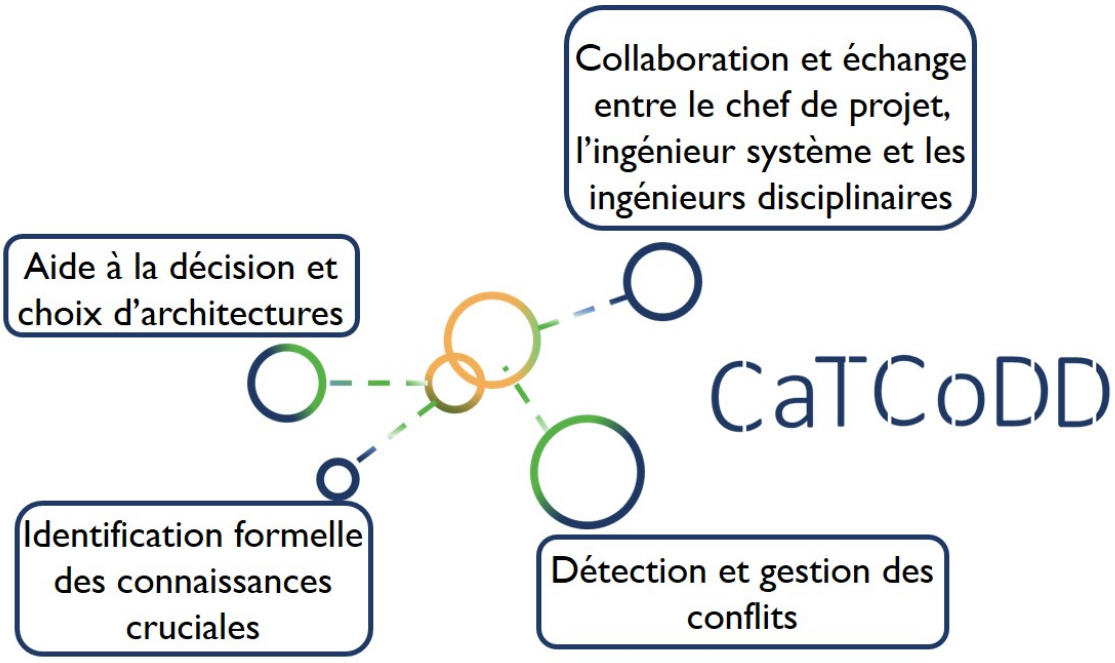


FIGURE I – Les principaux résultats de ce travail de recherche

Dans ces travaux de recherche, l'identification des connaissances cruciales et la résolution des conflits ont été basées sur la théorie des catégories. D'une part, cette théorie a fourni un cadre unifiant des modèles hétérogènes impliqués dans le proces-

sus de conception. Cette formalisation des connaissances nous a également permis d'éviter les échanges directs entre le chef de projet, l'ingénieur système et les ingénieurs disciplinaires afin de trouver un compromis entre eux et identifier les connaissances qui nécessitent la capitalisation ce qui peut être difficile à organiser dans un environnement distribué tel que celui de la conception collaborative mécatronique.

D'autre part, la localisation des conflits dans notre méthodologie CRP a été effectuée d'une manière formelle par le biais de foncteurs qui assurent une correspondance entre les différents objets ainsi que les morphismes de la catégorie des patterns vers la catégorie des paramètres. En outre, les avantages de l'emploi de la théorie des catégories peuvent être prouvés à long terme. En effet, la représentation des connaissances cruciales et les différentes instances facilitera la traçabilité de la collaboration. En assurant cette traçabilité il sera possible d'identifier qu'un paramètre donné a été utilisé et modifié par un expert donné à un moment donné. De plus, la traçabilité de l'évolution de la collaboration facilite les phases de réutilisation dans des projets de conception futurs.

En plus de l'identification formelle des connaissances et la gestion des conflits, notre méthodologie CaTCoDD nous a permis de comparer plusieurs architectures d'un système mécatronique dès les premières phases de conception. Cette comparaison précoce a un impact significatif sur le temps de développement ainsi que le coût du produit. Nous avons choisi d'intégrer la méthode d'aide à la décision multicritère AHP dans notre méthodologie générale grâce à son modèle compréhensible et souple pour résoudre les problèmes non structurés. Cette méthode est basée sur l'attribution des valeurs numériques à des jugements subjectifs et sur la synthèse de ces jugements pour déterminer les variables ayant la plus grande priorité.

Ainsi, il a été remarqué tout au long de ce mémoire que notre méthodologie CaTCoDD et son méta-modèle associé ont apporté une réponse à la problématique de collaboration et échange des données de granulartié fines entre différents domaines d'expertise, en offrant des solutions efficaces d'identification des connaissances qui nécessitent la capitalisation, de localisation et résolution des conflits et de choix d'architectures.

Perspectives de recherche

Les travaux de recherche menés dans cette thèse de doctorat ouvrent plusieurs perspectives. Tout d'abord, même si les deux systèmes mécatroniques utilisés durant la validation de notre approche CaTCoDD ont permis de valider les différentes étapes de notre proposition, nous pensons néanmoins qu'il serait intéressant de déployer la validation de notre méthodologie CaTCoDD sur d'autres systèmes complexes tels les systèmes autonomes (par exemple, la zipline pour livrer du sang en utilisant la technologie des drones et les systèmes cyber-physiques comme la pompe à insuline pour délivrer l'insuline aux diabétiques par perfusion continue), afin identifier des problèmes de collaboration encore non déclarés et mettre ainsi en valeur d'autres

domaines d'expertise.

Ensuite, en se basant sur le démonstrateur CaTCoDD développé durant ce travail de recherche nous avons eu une implémentation réelle de notre approche de conception collaborative et d'aide à la décision. Cependant, des tests sur des scénarios industriels dans un environnement de conception distribué pourraient apporter des retours d'expériences réels. Ces tests peuvent également clarifier les effets de l'application de CaTCoDD sur le rendement des entreprises qui ont adopté déjà une approche collaborative dans la conception des produits.

En ce qui concerne la gestion des conflits, seuls les incohérences entre les connaissances de granulartié fine (paramètres et contraintes) ont été prises en compte. Il serait intéressant d'étendre cette résolution à la gestion des incohérences de terminologie et de nom. Ainsi, il faudrait appliquer les méthodologies de synchronisation des modèles afin de proposer des solutions efficaces pour la gestion d'incohérences de nom et de terminologie.

D'autre part, la gestion des conflits dans notre approche CaTCoDD a été basée sur un certains nombre d'actions de résolution définies et prises par le chef de projet. De ce fait, il nous semble utile d'utiliser le concept de la satisfaction des contraintes ou (*Constraint Satisfaction Problem* CSP) afin de mieux tenir en considération les contraintes et simplifier ainsi la résolution des conflits.

Finalement, le choix d'architectures dans notre méthodologie a été effectué avec l'intégration de la méthode d'aide à la décision multicritère AHP afin de comparer les différentes architectures et obtenir ainsi un score global pour aider le chef de projet dans son choix. Ainsi, nous pensons à intégrer les approches de l'intelligence artificielle tels que la logique floue dans le processus de décision pour prendre en compte les facteurs d'imprécision et de subjectivité dans la comparaison entre les architectures.

Annexes

Annexe 1 : Extrait du programme MATLAB pour le choix d'architectures et l'aide à la décision

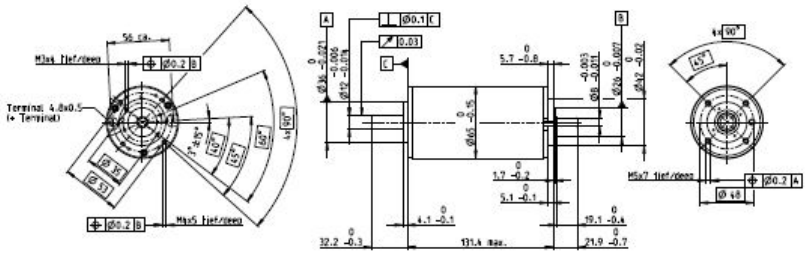
```
function pushbutton97_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton97 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.tab45,  'visible', 'off')
%% Input priorities of user
imp = (inputdlg({'Enter the importance of Criterion 1 over
   Criterion 1 on a scale of 1 to 9',...
    'Enter the importance of Criterion 1 over Criterion 2 on a
   scale of 1 to 9 ',...
    'Enter the importance of Criterion 1 over Criterion 3 on a
   scale of 1 to 9 ',...
    'Enter the importance of Criterion 2 over Criterion 1 on a
   scale of 1 to 9 ',...
    'Enter the importance of Criterion 2 over Criterion 2 on a
   scale of 1 to 9  ',...
    'Enter the importance of Criterion 2 over Criterion 3 on a
   scale of 1 to 9 ',...
    'Enter the importance of Criterion 3 over Criterion 1 on a
   scale of 1 to 9 ',...
    'Enter the importance of Criterion 3 over Criterion 2 on a
   scale of 1 to 9 ',...
    'Enter the importance of Criterion 3 over Criterion 3 on a
   scale of 1 to 9 '},...
    'Specify Importance of Criteria for AHP Matrix',[0.9 70]));
imp = str2double(imp);
ccmatrix=zeros(3,3);
ccmatrix(1,1)=imp(1);
ccmatrix(1,2)=imp(2);
ccmatrix(1,3)=imp(3);
ccmatrix(2,1)=imp(4);
ccmatrix(2,2)=imp(5);
ccmatrix(2,3)=imp(6);
ccmatrix(3,1)=imp(7);
ccmatrix(3,2)=imp(8);
ccmatrix(3,3)=imp(9);
```

```
f = figure;
t = uitable(f,'Data',ccmatrix,'Position',[20 300 308 77], '
    ColumnName',{'Rising time', 'Mass', 'Power'}, 'RowName',{'Rising
     time'; 'Mass'; 'Power'});
normmatrix=zeros(3,3);
normmatrix(1,1)=imp(1)/(imp(1)+ imp(4)+ imp(7));
normmatrix(2,1)=imp(4)/(imp(1)+ imp(4)+ imp(7));
normmatrix(3,1)=imp(7)/(imp(1)+ imp(4)+ imp(7));

normmatrix(1,2)=imp(2)/(imp(2)+ imp(5)+ imp(8));
normmatrix(2,2)=imp(5)/(imp(2)+ imp(5)+ imp(8));
normmatrix(3,2)=imp(8)/(imp(2)+ imp(5)+ imp(8));

normmatrix(1,3)=imp(3)/(imp(3)+ imp(6)+ imp(9));
normmatrix(2,3)=imp(6)/(imp(3)+ imp(6)+ imp(9));
normmatrix(3,3)=imp(9)/(imp(3)+ imp(6)+ imp(9));

  S = sum (normmatrix, 2);
  S1 = round(S, 2);
  X = S/3;
  X1 = round (X,2);

    lambdamax= (3.014 + 3.002 + 3.005)/3;
  input = (inputdlg({'Enter the number of criteria'},...
      'Specify number of criteria ' ,[1 30] ));
      input = str2double(input);

  CI = (lambdamax - input)/(input - 1);

  CIr = round (CI, 3);

         if (input == 3)
             RI= 0.58;
         elseif (input == 4)
             RI = 0.9;
         elseif (input == 5)
             RI = 1.12;
         elseif (input == 6)
             RI = 1.24;
         end

 CR = CIr/RI;

 if (CR < 0.10)
    msgbox('judgments matrix is reasonably consistent');
 else
    msgbox('judgments matrix is inconsistent, modify matrix');
 end
```

Annexe 2 : Caractéristiques du moteur DC sélectionné dans le modèle COTS pour la conception de l'EMA

RE 65 ∅65 mm, Graphite Brushes, 250 Watt

RE



M 1:4

Stock program
Standard program
Special program (on request)

		Part Numbers							
	Industrial Version IP54*	353294 388984	353295 388985	353296 388986	353297 388987	353298 388988	353299 388989	353300 388990	353301 388991
Motor Data									
	Values at nominal voltage								
1	Nominal voltage	V	18	24	36	48	60	70	70	70
2	No load speed	rpm	3520	4090	3970	3670	3680	3440	3190	2690
3	No load current	mA	755	697	437	289	231	179	160	125
4	Nominal speed	rpm	3250	3810	3700	3420	3450	3220	2960	2470
5	Nominal torque (max. continuous torque)	mNm	427	501	751	900	813	832	839	888
6	Nominal current (max. continuous current)	A	10	10	9.32	6.8	5.53	4.51	4.21	3.74
7	Stall torque	mNm	13800	15700	17400	16100	16200	15100	13700	12200
8	Stall current	A	295	292	207	131	106	78.6	66.1	49.7
9	Max. efficiency	%	81	83	87	88	89	89	89	89
	Characteristics									
10	Terminal resistance	Ω	0.0609	0.0821	0.174	0.365	0.568	0.891	1.06	1.41
11	Terminal inductance	mH	0.023	0.031	0.076	0.161	0.251	0.393	0.458	0.644
12	Torque constant	mNm/A	46	53.7	84.4	123	153	192	207	245
13	Speed constant	rpm/V	208	178	113	77.8	62.3	49.8	46.1	38.9
14	Speed / torque gradient	rpm/mNm	0.275	0.272	0.234	0.231	0.231	0.231	0.236	0.223
15	Mechanical time constant	ms	3.98	3.68	3.38	3.25	3.19	3.16	3.16	3.13
16	Rotor inertia	gcm²	1380	1290	1380	1340	1320	1310	1280	1340

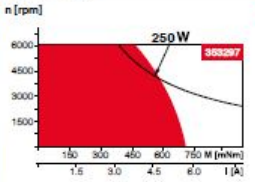
Specifications

	Thermal data	
17	Thermal resistance housing-ambient	1.3 K/W
18	Thermal resistance winding-housing	1.85 K/W
19	Thermal time constant winding	123 s
20	Thermal time constant motor	1060 s
21	Ambient temperature	-30...+100°C
22	Max. winding temperature	+125°C
	Mechanical data (preloaded ball bearings)	
23	Max. speed	5500 rpm
24	Axial play at axial load < 25 N	0 mm
	> 25 N	0.1 mm
25	Radial play	preloaded
26	Max. axial load (dynamic)	70 N
27	Max. force for press fits (static)	420 N
	(static, shaft supported)	12000 N
28	Max. radial load, 15 mm from flange	350 N
	Other specifications	
29	Number of pole pairs	2
30	Number of commutator segments	26
31	Weight of motor	2100 g

Values listed in the table are nominal.
Explanation of the figures on page 72.

* Industrial version with radial shaft seal ring (resulting in increased no load current). IP54 protection only if mounted on brush side, in compliance with maxon modular system.

Operating Range



Comments

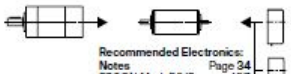
Continuous operation
In observation of above listed thermal resistance (lines 17 and 18) the maximum permissible winding temperature will be reached during continuous operation at 25°C ambient.
– Thermal limit.

Short term operation
The motor may be briefly overloaded (recurring).

Assigned power rating

maxon Modular System — Details on catalog page 34

Planetary Gearhead
∅81 mm
20 - 120 Nm
Page 404



Recommended Electronics:

Notes	Page 34
ESCON Mod. 50/5	487
ESCON Mod. 50/8 (HE)	488
ESCON 50/5	489
ESCON 70/10	489
EPOS4 Module 50/8	497
EPOS4 Module 50/15	497
EPOS4 Comp. 50/8 CAN	499
EPOS4 Comp. 50/15 CAN	500
EPOS4 70/15	501

Encoder HEDS 5540
500 CPT,
3 channels
Page 472

Encoder HEDL 5540
500 CPT,
3 channels
Page 474

Industrial Version IP54*

Encoder HEDL 9140
Page 479

Brake AB 44
Page 524

End cap
Page 525

Bibliographie

[1] Chen Zheng, Julien Le Duigou, Matthieu Bricogne, and Benoît Eynard. Multidisciplinary interface model for design of mechatronic systems. *Computers in Industry*, 76 :24–37, 2016.

[2] Davy Monticolo, Julien Badin, Samuel Gomes, Eric Bonjour, and Dominique Chamoret. A meta-model for knowledge configuration management to support collaborative engineering. *Computers in Industry*, 66 :11–20, 2015.

[3] Jonas Mørkeberg Torry-Smith, Ahsan Qamar, Sofiane Achiche, Jan Wikander, Niels Henrik Mortensen, and Carl During. Challenges in designing mechatronic systems. *Journal of Mechanical Design*, 135(1), 2013.

[4] Chen Zheng, Matthieu Bricogne, Julien Le Duigou, and Benoit Eynard. Survey on mechatronic engineering : A focus on design methods and product models. *Advanced Engineering Informatics*, 28(3) :241–257, 2014.

[5] Virginie Fortineau, Thomas Paviot, and Samir Lamouri. Improving the interoperability of industrial information systems with description logic-based models—the state of the art. *Computers in Industry*, 64(4) :363–375, 2013.

[6] Vijay Srinivasan. An integration framework for product lifecycle management. *Computer-aided design*, 43(5) :464–478, 2011.

[7] Mehdi Mcharek, Moncef Hammadi, Toufik Azib, Cherif Larouci, and Jean-Yves Choley. Collaborative design process and product knowledge methodology for mechatronic systems. *Computers in Industry*, 105 :213–228, 2019.

[8] Mehdi Mcharek, Toufik Azib, Moncef Hammadi, Cherif Larouci, and Jean-Yves Choley. Multidisciplinary design optimization using knowledge management applied to an electronic throttle. *COMPEL-The international journal for computation and mathematics in electrical and electronic engineering*, 2020.

[9] Inès Saad and Salem Chakhar. A decision support for identifying crucial knowledge requiring capitalizing operation. *European Journal of operational research*, 195(3) :889–904, 2009.

[10] Julien Badin, Dominique Chamoret, Samuel Gomes, and Davy Monticolo. Knowledge configuration management for product design and numerical simulation. In *DS 68-6 : Proceedings of the 18th International Conference on*

Engineering Design (ICED 11), Impacting Society through Engineering Design, Vol. 6 : Design Information and Knowledge, Lyngby/Copenhagen, Denmark, 15.-19.08. 2011, 2011.

[11] Michel Grundstein. From capitalizing on company knowledge to knowledge management. *Knowledge management, classic and contemporary works*, 12 :261–287, 2000.

[12] Stefan Feldmann, Manuel Wimmer, Konstantin Kernschmidt, and Birgit Vogel-Heuser. A comprehensive approach for managing inter-model inconsistencies in automated production systems engineering. In *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 1120–1127. IEEE, 2016.

[13] Weslley Torres, Mark van den Brand, and Alexander Serebrenik. Model management tools for models of different domains : a systematic literature review. In *2019 IEEE International Systems Conference (SysCon)*, pages 1–8. IEEE, 2019.

[14] Eleftherios Katrantzis, Vassilis C Moulianitis, and Kanstantsin Miatliuk. Conceptual design evaluation of mechatronic systems. In *Emerging Trends in Mechatronics*, page 27. IntechOpen, 2020.

[15] Robert H Bishop and Robert H Bishop. *The mechatronics handbook*, volume 121. CRC press Boca Raton, 2002.

[16] Nobuhiro Kyura and Hirosuke Oho. Mechatronics-an industrial perspective. *IEEE/ASME transactions on mechatronics*, 1(1) :10–15, 1996.

[17] Fumio Harashima, Masayoshi Tomizuka, and Toshio Fukuda. Mechatronics-" what is it, why, and how ?" an editorial. *IEEE/ASME Transactions on Mechatronics*, 1(1) :1–4, 1996.

[18] Rolf Isermann. *Mechatronic systems : fundamentals*. Springer Science & Business Media, 2007.

[19] Dean C Karnopp, Donald L Margolis, and Ronald C Rosenberg. *System dynamics : modeling, simulation, and control of mechatronic systems*. John Wiley & Sons, 2012.

[20] Audrey Jardin. *Contribution à une méthodologie de dimensionnement des systèmes mécatroniques : analyse structurelle et couplage à l'optimisation dynamique*. PhD thesis, INSA de Lyon, 2010.

[21] Baris Canbaz. *Preventing and resolving design conflicts for a collaborative convergence in distributed set-based design*. PhD thesis, Ecole Centrale Paris, 2013.

[22] Gerhard Pahl and Wolfgang Beitz. *Engineering design : a systematic approach*. Springer Science & Business Media, 2013.

[23] Joaquin Aca, Marcopolo Ramos, Jose L Serrano, Horacio Ahuett, and Arturo Molina. Concurrent engineering of mechatronic products in virtual enterprises : selection and deployment of a plm system for the machine tool industry. In *International Conference on Cooperative Design, Visualization and Engineering*, pages 318–326. Springer, 2006.

[24] Matthieu Bricogne, Nadège Troussier, Louis Rivest, and Benoît Eynard. Agile design methods for mechatronics system integration. In *IFIP International Conference on Product Lifecycle Management*, pages 458–470. Springer, 2013.

[25] VDI-Fachbereich Produktentwicklung und Mechatronik. Design methodology for mechatronic systems. *VDI-Gesellschaft Produktund Prozessgestaltung, Norme VDI*, 2206, 2004.

[26] Stefan Wiesner, Mike Freitag, Ingo Westphal, and Klaus-Dieter Thoben. Interactions between service and product lifecycle management. *Procedia Cirp*, 30 :36–41, 2015.

[27] Martin Törngren, Ahsan Qamar, Matthias Biehl, Frederic Loiret, and Jad El-Khoury. Integrating viewpoints in the development of mechatronic products. *Mechatronics*, 24(7) :745–762, 2014.

[28] Moncef Hammadi, Jean-Yves Choley, Olivia Penas, and Alain Riviere. Multidisciplinary approach for modelling and optimization of road electric vehicles in conceptual design level. In *2012 Electrical Systems for Aircraft, Railway and Ship Propulsion*, pages 1–6. IEEE, 2012.

[29] Jens Bathelt, Anders Jönsson, Christian Bacs, Stefan Dierssen, Markus Meier, et al. Applying the new vdi design guideline 2206 on mechatronic systems controlled by a plc. In *DS 35 : Proceedings ICED 05, the 15th International Conference on Engineering Design, Melbourne, Australia, 15.-18.08. 2005*, pages 415–416, 2005.

[30] Iris Graessler and Julian Hentze. The new v-model of vdi 2206 and its validation. *at-Automatisierungstechnik*, 68(5) :312–324, 2020.

[31] Peter Hehenberger, F Poltschak, Klaus Zeman, and W Amrhein. Hierarchical design models in the mechatronic product development process of synchronous machines. *Mechatronics*, 20(8) :864–875, 2010.

[32] Mogens Myrup Andreasen, AHB Duffy, KJ MacCallum, J Bowen, and T Storm. The design co-ordination framework : key elements for effective product development. In *The design productivity debate*, pages 151–172. Springer, 1998.

[33] Biren Prasad. Enabling principles of concurrency and simultaneity in concurrent engineering. *AI EDAM*, 13(3) :185–204, 1999.

[34] SC-Y Lu, Waguih ElMaraghy, Günther Schuh, and Robert Wilhelm. A scientific foundation of collaborative engineering. *CIRP annals*, 56(2) :605–634, 2007.

[35] Frédéric Demoly, Davy Monticolo, Benoît Eynard, Louis Rivest, and Samuel Gomes. Multiple viewpoint modelling framework enabling integrated product–process design. *International Journal on Interactive Design and Manufacturing (IJIDeM)*, 4(4) :269–280, 2010.

[36] Frédéric Demoly. *Conception intégrée et gestion d'informations techniques : application à l'ingénierie du produit et de sa séquence d'assemblage.* PhD thesis, Université de Technologie de Belfort-Montbeliard, 2010.

[37] Cecilia Haskins. Incose systems engineering handbook : A guide for sytem life cycle processes and activities. Incose, 2007.

[38] Faïda Mhenni, Jean-Yves Choley, Olivia Penas, Régis Plateaux, and Moncef Hammadi. A sysml-based methodology for mechatronic systems architectural design. *Advanced Engineering Informatics*, 28(3) :218–231, 2014.

[39] ANSI/EIA. Ansi/eia-632-1988 processes for engineering a system, 1999.

[40] ISO/IEC/IEEE. Iso/iec/ieee 24748-2 :2018 systems and software engineering — life cycle management — part 2 : Guidelines for the application of iso/iec/ieee 15288 (system life cycle processes), 2018. Last accessed 06 June 2021.

[41] Peter Hehenberger and David Bradley. *Mechatronic futures : challenges and solutions for mechatronic systems and their designers.* Springer, 2016.

[42] Aude Warniez, Olivia Penas, Régis Plateaux, and Romain Barbedienne. Sysml geometrical profile development for physical integration of mechatronic systems. In *14th Mechatronics Forum International Conference, Mechatronics 2014*, 2014.

[43] OMG SysML. Systems modeling language. *OMG [online](Available :*, 2006.

[44] Julien Badin. *Ingénierie hautement productive et collaborative à base de connaissances métier : vers une méthodologie et un méta-modèle de gestion des connaissances en configurations.* PhD thesis, Belfort-Montbéliard, 2011.

[45] Kleanthis Thramboulidis et al. The 3+ 1 sysml view-model in model integrated mechatronics. *Journal of Software Engineering and Applications*, 3(02) :109, 2010.

[46] Manas Bajaj, Dirk Zwemer, Russell Peak, Alex Phung, Andrew Scott, and Miyako Wilson. 4.3. 1 satellites to supply chains, energy to finance—slim for model-based systems engineering : Part 1 : Motivation and concept of slim. In *INCOSE international Symposium*, volume 21, pages 368–394. Wiley Online Library, 2011.

[47] Mohammad Chami and Jean-Michel Bruel. Towards an integrated conceptual design evaluation of mechatronic systems : The sysdice approach. *Procedia Computer Science*, 51 :650–659, 2015.

[48] Tania Tudorache. Employing ontologies for an improved development process in collaborative engineering. 2006.

[49] Luyan Bi, Zongxia Jiao, and Shengtao Fan. Ontology-based information integration framework for mechatronics system multi-disciplinary design. In *2008 6th IEEE International Conference on Industrial Informatics*, pages 831–836. IEEE, 2008.

[50] Haibo Hong, Yuehong Yin, and Xing Chen. Ontological modelling of knowledge management for human–machine integrated design of ultra-precision grinding machine. *Enterprise Information Systems*, 10(9) :970–981, 2016.

[51] Fa Lin Wang, Yu Guo, Wen He Liao, and Bao Sheng Wu. Knowledge push technology for complex mechatronic products design based on ontology and variable precision rough set. In *Applied Mechanics and Materials*, volume 799, pages 1107–1112. Trans Tech Publ, 2015.

[52] Régis Plateaux, Olivia Penas, Peter Hehenberger, Moncef Hammadi, Faïda Mhenni, Aude Warniez, and Jean-Yves Choley. Needs for a 3d enriched ontology for mechatronic systems design. In *2015 IEEE International Symposium on Systems Engineering (ISSE)*, pages 253–260. IEEE, 2015.

[53] Emergent Systems. Emergent systems - enterprise engineering knowledge system, (n.d.), 2010. Last accessed 09 June 2021.

[54] Sven Kleiner, Reiner Anderl, and Robert Gräb. A collaborative design system for product data integration. *Journal of engineering design*, 14(4) :421–428, 2003.

[55] Farouk Belkadi, Nicolas Dremont, Alban Notin, Nadege Troussier, and Mourad Messadia. A meta-modelling framework for knowledge consistency in collaborative design. *Annual Reviews in Control*, 36(2) :346–358, 2012.

[56] Mehdi Mcharek, Toufik Azib, Moncef Hammadi, Jean-Yves Choley, and Cherif Larouci. Knowledge sharing for mechatronic systems design and optimization. *IFAC-PapersOnLine*, 51(11) :1365–1370, 2018.

[57] Nicolas Dremont. *Maturity integrated in a meta model of knowledge to help decision making in preliminary collaborative design of mechanical systems*. PhD thesis, Compiègne, 2013.

[58] Mouna Fradi, Raoudha Gaha, Abdelfattah Mlika, Faïda Mhenni, and Jean Yves Choley. Design of an electronic throttle body based on a new knowledge sharing engineering methodology. In *International Conference Design and Modeling of Mechanical Systems*, pages 55–63. Springer, 2019.

[59] Tzu-Liang Tseng and Chun-Che Huang. Capitalizing on knowledge : a novel approach to crucial-knowledge determination. *IEEE Transactions on Systems, Man, and Cybernetics-Part A : Systems and Humans*, 35(6) :919–931, 2005.

[60] Michel Grundstein. De la capitalisation des connaissances au management des connaissances dans l'entreprise. 2004.

[61] Michel Grundstein and Camille Rosenthal-Sabroux. Gameth®, a decision support approach to identify and locate potential crucial knowledge. In *Proceedings 5th European Conference on Knowledge Management*, pages 391–402. Citeseer, 2004.

[62] Joanna Pomian and Claude Roche. *Connaissance capitale : Management des connaissances et organisation du travail*. Editions L'Harmattan, 2002.

[63] Jean-Louis Ermine. A theoretical and formal model for knowledge management systems. In *ICICKM'2005 : 2nd International Conference on Intellectual Capital and Knowledge Management*, pages 187–199, 2005.

[64] Samar Ammar-Khodja, Nicolas Perry, and Alain Bernard. Processing knowledge to support knowledge-based engineering systems specification. *Concurrent Engineering*, 16(1) :89–101, 2008.

[65] Imène Brigui-Chtioui and Inès Saad. A multiagent approach for collective decision making in knowledge management. *Group Decision and Negotiation*, 20(1) :19–37, 2011.

[66] Sahar Ghrab, Ines Saad, Faiez Gargouri, and Gilles Kassel. A decision support system for identifying and representing likely crucial organizational know-how and knowing that. *Journal of decision systems*, 23(3) :266–284, 2014.

[67] Mariam Ben Hassen, Mohamed Turki, and Faïez Gargouri. A multi-criteria evaluation approach for selecting a sensitive business process modeling language for knowledge management. *Journal on Data Semantics*, 8(3) :157–202, 2019.

[68] Nenad Medvidovic and Richard N Taylor. Software architecture : foundations, theory, and practice. In *2010 ACM/IEEE 32nd International Conference on Software Engineering*, volume 2, pages 471–472. IEEE, 2010.

[69] Jakob Axelsson. Achieving system-of-systems interoperability levels using linked data and ontologies. In *INCOSE International Symposium*, volume 30, pages 651–665. Wiley Online Library, 2020.

[70] Christophe Guychard, Sylvain Guerin, Ali Koudri, Antoine Beugnard, and Fabien Dagnat. Conceptual interoperability through models federation. In *Semantic Information Federation Community Workshop*, page 23, 2013.

[71] Ahsan Qamar, Matthew Meinhart, and George Walley. Model based systems engineering to support failure mode avoidance for driver-assistance systems. In *2017 IEEE Aerospace Conference*, pages 1–9. IEEE, 2017.

[72] Conrad Bock, XuanFang Zha, Hyo-won Suh, and Jae-Hyun Lee. Ontological product modeling for collaborative design. *Advanced Engineering Informatics*, 24(4) :510–524, 2010.

[73] Olivia Penas, Régis Plateaux, Stanislao Patalano, and Moncef Hammadi. Multi-scale approach from mechatronic to cyber-physical systems for the design of manufacturing systems. *Computers in Industry*, 86 :52–69, 2017.

[74] Ahsan Qamar, Christiaan JJ Paredis, Jan Wikander, and Carl During. Dependency modeling and model management in mechatronic design. *Journal of Computing and Information Science in Engineering*, 12(4), 2012.

[75] Anthony Legendre, Agnes Lanusse, and Antoine Rauzy. Toward model synchronization between safety analysis and system architecture design in industrial contexts. In *International Symposium on Model-Based Safety and Assessment*, pages 35–49. Springer, 2017.

[76] Aroua Berriche, Faïda Mhenni, Abdelfattah Mlika, and Jean-Yves Choley. Towards model synchronization for consistency management of mechatronic systems. *Applied Sciences*, 10(10) :3577, 2020.

[77] Stefan Feldmann, Konstantin Kernschmidt, Manuel Wimmer, and Birgit Vogel-Heuser. Managing inter-model inconsistencies in model-based systems engineering : Application in automated production systems engineering. *Journal of Systems and Software*, 153 :105–134, 2019.

[78] Pierre-Alain Yvars. A csp approach for the network of product lifecycle constraints consistency in a collaborative design context. *Engineering applications of artificial intelligence*, 22(6) :961–970, 2009.

[79] Baris Canbaz, Bernard Yannou, and Pierre-Alain Yvars. Preventing design conflicts in distributed design systems composed of heterogeneous agents. *Engineering Applications of Artificial Intelligence*, 28 :142–154, 2014.

[80] Baris Canbaz, Bernard Yannou, and Pierre-Alain Yvars. Resolving design conflicts and evaluating solidarity in distributed design. *IEEE Transactions on Systems, Man, and Cybernetics : Systems*, 44(8) :1044–1055, 2014.

[81] Heng Kuang. *Towards a formal reactive autonomic systems framework using category theory*. PhD thesis, Concordia University, 2013.

[82] Olga Ormandjieva, Jamal Bentahar, Jinzi Huang, and Heng Kuang. Modelling multi-agent systems with category theory. *Procedia Computer Science*, 52 :538–545, 2015.

[83] Bernard Roy. Decision-aid and decision-making. *European Journal of Operational Research*, 45(2-3) :324–331, 1990.

[84] Pierre Couturier, Mambaye Lô, Abdelhak Imoussaten, Vincent Chapurlat, and Jacky Montmain. Tracking the consequences of design decisions in mechatronic systems engineering. *Mechatronics*, 24(7) :763–774, 2014.

[85] VC Moulianitis, NA Aspragathos, and AJ Dentsoras. A model for concept evaluation in design—-an application to mechatronics design of robot grippers. *Mechatronics*, 14(6) :599–622, 2004.

[86] Saeed Behbahani and Clarence W de Silva. Mechatronic design quotient as the basis of a new multicriteria mechatronic design methodology. *IEEE/ASME Transactions on mechatronics*, 12(2) :227–232, 2007.

[87] Moncef Hammadi, JY Choley, Olivia Penas, Alain Riviere, Jamel Louati, and Mohamed Haddar. A new multi-criteria indicator for mechatronic system performance evaluation in preliminary design level. In *2012 9th France-Japan & 7th Europe-Asia Congress on Mechatronics (MECATRONICS)/13th Int'l Workshop on Research and Education in Mechatronics (REM)*, pages 409–416. IEEE, 2012.

[88] Abolfazl Mohebbi, Sofiane Achiche, and Luc Baron. Mechatronic multicriteria profile (mmp) for conceptual design of a robotic visual servoing system. In *Engineering Systems Design and Analysis*, volume 45851, page V003T15A015. American Society of Mechanical Engineers, 2014.

[89] VC Moulianitis, G-AD Zachiotis, and NA Aspragathos. A new index based on mechatronics abilities for the conceptual design evaluation. *Mechatronics*, 49 :67–76, 2018.

[90] Ruirui Chen, Yusheng Liu, Hongri Fan, Jianjun Zhao, and Xiaoping Ye. An integrated approach for automated physical architecture generation and multi-criteria evaluation for complex product design. *Journal of Engineering Design*, 30(2-3) :63–101, 2019.

[91] Roseanna W Saaty. The analytic hierarchy process—what it is and how it is used. *Mathematical modelling*, 9(3-5) :161–176, 1987.

[92] Ward Edwards. Social utilities. *Engineering Economist*, 6 :119–129, 1971.

[93] Máximo Méndez, Blas Galván, Daniel Salazar, and David Greiner. Multiple-objective genetic algorithm using the multiple criteria decision making method topsis. In *Multiobjective Programming and Goal Programming*, pages 145–154. Springer, 2009.

[94] Ralph L Keeney and Gary L Lilien. New industrial product design and evaluation using multiattribute value analysis. *Journal of Product Innovation Management : AN INTERNATIONAL PUBLICATION OF THE PRODUCT DEVELOPMENT & MANAGEMENT ASSOCIATION*, 4(3) :185–198, 1987.

[95] Berthier Roy and Patrice Bertier. *La méthode ELECTRE II : une méthode de classement en prédence de critères multiples*. 1971.

[96] Jean-Pierre Brans, Ph Vincke, and Bertrand Mareschal. How to select and how to rank projects : The promethee method. *European journal of operational research*, 24(2) :228–238, 1986.

[97] Adel Guitouni, Jean-Marc Martel, Philippe Vincke, et al. *Un cadre de référence pour le choix d'une procédure d'agrégation multicritère*. Faculté des sciences de l'administration de l'Université Laval, Direction de . . . , 1999.

[98] Mohamed A Mabrok and Michael J Ryan. Category theory as a formal mathematical foundation for model-based systems engineering. *Appl. Math. Inf. Sci*, 11 :43–51, 2017.

[99] David I Spivak. Category theory for scientists. *arXiv preprint arXiv :1302.6946*, 2013.

[100] Steven Roman. *An introduction to the language of category theory*. Springer, 2017.

[101] Yaniv Mordecai, James Fairbanks, and Edward F Crawley. Category-theoretic formulation of the model-based systems architecting cognitive-computational cycle. *Applied Sciences*, 11(4) :1945, 2021.

[102] Ming Zhu, Heng Kuang, and Jing Li. Representation of categorical specification of self-configurations in reactive autonomic systems framework. *Journal of Computer and Communications*, 6(12) :34–48, 2018.

[103] Ming Zhu and Jing Li. Towards a categorical framework for verifying design and implementation of concurrent systems. *Journal of Computer and Communications*, 6(11) :227–246, 2018.

[104] Nadew Kibret, William Edmonson, and Solomon Gebreyohannes. Category theoretic based formalization of the verifiable design process. In *2019 IEEE International Systems Conference (SysCon)*, pages 1–8. IEEE, 2019.

[105] Ron Sanchez and Aimé Heene. Reinventing strategic management : New theory and practice for competence-based competition. *European Management Journal*, 15(3) :303–317, 1997.

[106] Ikujiro Nonaka and Hirotaka Takeuchi. *The knowledge-creating company : How Japanese companies create the dynamics of innovation*. Oxford university press, 1995.

[107] Michel Grundstein. La capitalisation des connaissances de l'entreprise, système de production des connaissances. In *Actes du Colloque L'Entreprise Apprenante et les Sciences de la Complexité, Aix en Provence, France*, volume 22, page 24, 1995.

[108] Sameer Kumar. A knowledge based reliability engineering approach to manage product safety and recalls. *Expert Systems with Applications*, 41(11) :5323–5339, 2014.

[109] Maryam Alavi and Dorothy E Leidner. Knowledge management and knowledge management systems : Conceptual foundations and research issues. *MIS quarterly*, pages 107–136, 2001.

[110] Gianfranco La Rocca. Knowledge based engineering : Between ai and cad. review of a language based technology to support engineering design. *Advanced engineering informatics*, 26(2) :159–179, 2012.

[111] Jean-Louis Ermine. *La gestion des connaissances*. Hermès sciences publications, 2003.

[112] Amit Fisher, Mike Nolan, Sanford Friedenthal, Michael Loeffler, Mark Sampson, Manas Bajaj, Lonnie VanZandt, Krista Hovey, John Palmer, and Laura Hart. 3.1. 1 model lifecycle management for mbse. In *INCOSE International Symposium*, volume 24, pages 207–229. Wiley Online Library, 2014.

[113] Diana Penciuc, Alexandre Durupt, Farouk Belkadi, Benoît Eynard, and Harvey Rowson. Towards a plm interoperability for a collaborative design support system. *Procedia Cirp*, 25 :369–376, 2014.

[114] Sebastian JI Herzig, Ahsan Qamar, and Christiaan JJ Paredis. An approach to identifying inconsistencies in model-based systems engineering. *Procedia Computer Science*, 28 :354–362, 2014.

[115] Mohamed Firas Borchani, Moncef Hammadi, Noureddine Ben Yahia, and Jean-Yves Choley. Integrating model-based system engineering with set-based concurrent engineering principles for reliability and manufacturability analysis of mechatronic products. *Concurrent Engineering*, 27(1) :80–94, 2019.

[116] Mouna Fradi, Raoudha Gaha, Faïda Mhenni, Abdelfattah Mlika, and Jean-Yves Choley. Knowledge capitalization in mechatronic collaborative design. *Concurrent Engineering*, 2021.

[117] B Ashok, S Denis Ashok, and C Ramesh Kumar. Trends and future perspectives of electronic throttle control system in a spark ignition engine. *Annual Reviews in Control*, 44 :97–115, 2017.

[118] Mehdi Mcharek, Moncef Hammadi, Jean-Yves Choley, Toufik Azib, and Cherif Larouci. Modeling and multi-objective optimization of an electronic throttle in open-loop. In *2016 11th France-Japan & 9th Europe-Asia Congress on Mechatronics (MECATRONICS)/17th International Conference on Research and Education in Mechatronics (REM)*, pages 331–335. IEEE, 2016.

[119] Maxon. Maxon group, 2020. Last accessed 20 October 2020.

[120] Mouna Fradi, Faïda Mhenni, Raoudha Gaha, Abdelfattah Mlika, and Jean-Yves Choley. Conflict resolution in mechatronic collaborative design using category theory. *Applied Sciences*, 11(10) :4486, 2021.

[121] OMG MDA. Object management group model driven architecture, 2008.

[122] Sebastian JI Herzig. *A Bayesian learning approach to inconsistency identification in model-based systems engineering*. PhD thesis, Georgia Institute of Technology, 2015.

[123] Mouna Fradi, Faïda Mhenni, Raoudha Gaha, Abdelfattah Mlika, and Jean-Yves Choley. Conflict management for mechatronic systems design. In *2021 IEEE International Systems Engineering Symposium (ISSE)*. IEEE, 2021.

[124] Sebastian JI Herzig and Christiaan JJ Paredis. A conceptual basis for inconsistency management in model-based systems engineering. *Procedia CIRP*, 21 :52–57, 2014.

[125] Marija Jankovic, Julie Stal-Le Cardinal, and Jean-Claude Bocquet. Collaborative decision-making in design project management. a particular focus on automotive industry. *Journal of decision systems*, 19(1) :93–116, 2010.

[126] Michael Whelton, Glenn Ballard, and Iris D Tommelein. A knowledge management framework for project definition. *Journal of Information Technology in Construction (ITcon)*, 7(13) :197–212, 2003.

[127] Claus Thorp Hansen, Mogens Myrup Andreasen, et al. A mapping of design decision-making. In *DS 32 : Proceedings of DESIGN 2004, the 8th International Design Conference, Dubrovnik, Croatia*, pages 1409–1418, 2004.

[128] Serge Bellut. *La compétitivité par la maîtrise des coûts : Conception à coût objectif et analyse de la valeur*. Afnor, 1990.

[129] José Eugenio Leal. Ahp-express : A simplified version of the analytical hierarchy process method. *MethodsX*, 7 :100748, 2020.

[130] Ashu Gupta, KAWALJEET Singh, and Rajesh Verma. A critical study and comparison of manufacturing simulation softwares using analytic hierarchy process. *Journal of Engineering Science and Technology*, 5(1) :108–129, 2010.

[131] Enrique Mu and Milagros Pereyra-Rojas. Understanding the analytic hierarchy process. In *Practical decision making*, pages 7–22. Springer, 2017.

[132] Aurelien Reysset. *Conception préliminaire d'actionneurs électromécaniques-outils d'aide à la spécification et à la génération de procédures de dimensionnement pour l'optimisation*. PhD thesis, Toulouse, INSA, 2015.

[133] J Liscouët, J-C Maré, and Marc Budinger. An integrated methodology for the preliminary design of highly reliable electromechanical actuators : Search for architecture solutions. *Aerospace Science and Technology*, 22(1) :9–18, 2012.

[134] Jian Fu, Jean-Charles Maré, and Yongling Fu. Modelling and simulation of flight control electromechanical actuators with special focus on model architecting, multidisciplinary effects and power flows. *Chinese Journal of Aeronautics*, 30(1) :47–65, 2017.

[135] Guan Qiao, Geng Liu, Zhenghong Shi, Yawen Wang, Shangjun Ma, and Teik C Lim. A review of electromechanical actuators for more/all electric aircraft systems. *Proceedings of the Institution of Mechanical Engineers, Part C : Journal of Mechanical Engineering Science*, 232(22) :4128–4151, 2018.

[136] Enrique J Vidal and Elizabeth R Villota. Sysml as a tool for requirements traceability in mechatronic design. In *Proceedings of the 2018 4th International Conference on Mechatronics and Robotics Engineering*, pages 146–152, 2018.

[137] Delphine Mami. *Définition, conception et expérimentation de structures d'actionneurs électromécaniques innovants incluant par conception des fonctionnalités de sûreté et de sécurité de fonctionnement.* PhD thesis, 2010.

[138] Faïda Mhenni. *Safety analysis integration in a systems engineering approach for mechatronic systems design.* PhD thesis, Ecole Centrale Paris, 2014.

[139] Faïda Mhenni, Jean-Yves Choley, Françoise Caron, and Antoine Munck. Collaborative mechatronic design and systems engineering : an educational experiment with karren. In *2018 IEEE International Systems Engineering Symposium (ISSE)*, pages 1–7. IEEE, 2018.

[140] Peter Hehenberger, Birgit Vogel-Heuser, D Bradley, Benoît Eynard, Tetsuo Tomiyama, and Sofiane Achiche. Design, modelling, simulation and integration of cyber physical systems : Methods and applications. *Computers in Industry*, 82 :273–289, 2016.

[141] Hana Siala, Faïda Mhenni, Jean-Yves Choley, Maher Barkallah, Jamel Louati, and Mohamed Haddar. Toward a robust design of an aileron electromechanical actuator : Sensitivity analysis and parametric tolerancing using a variational approach. *IEEE Systems Journal*, 14(3) :3977–3986, 2020.

[142] Bhargav Dave and Lauri Koskela. Collaborative knowledge management—a construction case study. *Automation in construction*, 18(7) :894–902, 2009.

Liste des publications

Fradi M, Mhenni F, Gaha R, Mlika A, Choley JY, (2021), Category theory-based collaborative design methodology for mechatronic systems. Soumis dans une revue internationale avec comité de lecture intitulée : Advanced Engineering Informatics, Elseiver.

Fradi M, Gaha R, Mhenni F, Mlika A, Choley JY, (2021), Knowledge capitalization in mechatronic collaborative design. Accepté dans une revue internationale avec comité de lecture intitulée : Concurrent Engineering : research and application, Sage journals.

Fradi M, Mhenni F, Gaha R, Mlika A, Choley JY, (2021), Conflict resolution in mechatronic collaborative design using category theory. Accepté dans une revue internationale avec comité de lecture intitulée : Applied sciences, MDPI publisher.

Fradi M, Mhenni F, Gaha R, Mlika A, Choley JY, (2021), Conflict management for mechatronic systems design. In 2021 IEEE International Systems Engineering Symposium (ISSE). IEEE.

Fradi M, Gaha R, Mlika A, Mhenni F, Choley JY, (2019), Survey on mechatronic collaborative design : A focus on product and knowledge models and decision-making methods. In International Congress of Mechanics, Mechatronics and Materials Design and Modeling of Mechanical Systems (IC3M'2019).

Fradi M, Gaha R, Mlika A, Mhenni F, Choley JY, (2019), Design of an electronic throttle body based on a new knowledge sharing engineering methodology. In International Conference Design and Modeling of Mechanical Systems, pages 55–63. Springer.