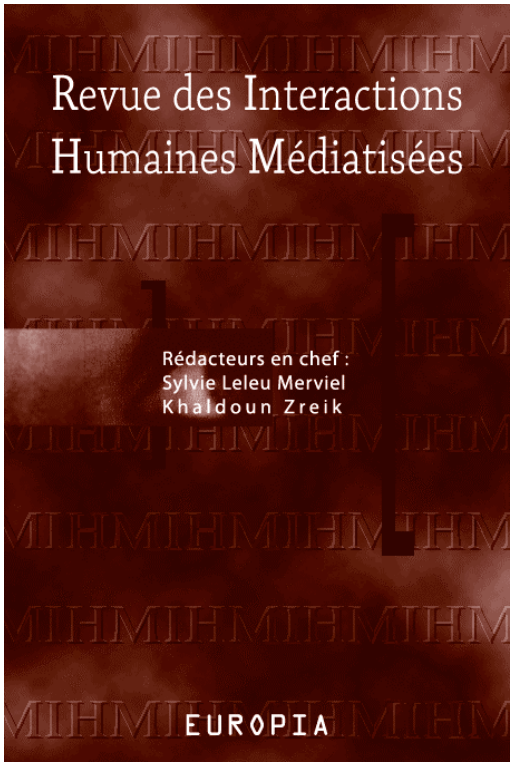
Revue des Interactions Humaines Médiatisées

Journal of Human Mediated Interactions

Rédacteurs en chef

Sylvie Leleu-Merviel & Khaldoun Zreik

Vol 17 - N° 2 / 2016



© europia, 2017
15, avenue de Ségur,
75007 Paris - France
http://europia.org/RIHM
rihm@europia.org

Revue des Interactions Humaines Médiatisées

Journal of Human Mediated Interactions

Rédacteurs en chef / *Editors in chief*

- Sylvie Leleu-Merviel, Université de Valenciennes et du Hainaut-Cambrésis, Laboratoire DeVisu
- Khaldoun Zreik, Université Paris 8, Laboratoire Paragraphe

Comité éditorial / *Editorial Board*

- Thierry Baccino (Université Paris8, LUTIN - UMS-CNRS 2809, France)
- Karine Berthelot-Guiet (CELSA- Paris-Sorbonne GRIPIC, France)
- Pierre Boulanger (University of Alberta, Advanced Man-Machine Interface Laboratory, Canada)
- Jean-Jacques Boutaud (Université de Dijon, CIMEOS, France)
- Aline Chevalier (Université Paris Ouest Nanterre La Défense, CLLE-LTC, France)
- Yves Chevalier (Université de Bretagne Sud, CERSIC -ERELLIF, France)
- Didier Courbet (Université de la Méditerranée Aix-Marseille II, Mediasic, France)
- Viviane Couzinet (Université de Toulouse3, LERASS, France)
- Milad Doueihi (Université de Laval - Chaire de recherche en Cultures numériques, Canada)
- Pierre Fastrez (Université Catholique de Louvain, GReMS, Belgique)
- Pascal Francq (Université Catholique de Louvain, ISU, Belgique)
- Bertrand Gervais (UQAM, Centre de Recherche sur le texte et l'imaginaire, Canada)
- Yves Jeanneret (CELSA- Paris-Sorbonne GRIPIC, France)
- Patrizia Laudati (Université de Valenciennes, DeVisu, France)
- Catherine Loneux (Université de Rennes, CERSIC -ERELLIF, France)
- Marion G. Müller (Jacobs University Bremen, PIAV, Allemagne)
- Marcel O'Gormann (Univerity of Waterloo, Critical Média Lab, Canada)
- Serge Proulx (UQAM, LabCMO, Canada)
- Jean-Marc Robert (Ecole Polytechnique de Montréal, Canada)
- Imad Saleh (Université Paris 8, CITU-Paragraphe, France)
- André Tricot (Université de Toulouse 2, CLLE - Lab. Travail & Cognition, France)
- Jean Vanderdonckt (Université Catholique de Louvain, LSM, Blgique)
- Alain Trognon (Université Nancy2, Laboratoire InterPsy, France)

Revue des Interactions Humaines Médiatisées

Journal of Human Mediated Interactions

Vol 17 - N°2 / 2016

Sommaire

Editorial
Sylvie LELEU-MERVIEL, Khaldoun ZREIK (Rédacteurs en chef) 1

WoDoMoLEID : un langage de modélisation de domaines de travail pour la conception d'interfaces écologiques
WoDoMoLEID: Work Domain Modeling Language for Ecological Interface Design
Alexandre MOÏSE (1), Jean-Marc ROBERT 3

REMIND : une méthode pour comprendre la micro-dynamique de l'expérience des visiteurs de musées
REMIND : a method to understand the micro-dynamics of the museum visitors experience
Daniel SCHMITT (1), Olivier AUBERT 43

Supports technologiques collaboratifs à la méthode des Personas
Collaborative technological supports to assist Persona method
Jessy BARRE, Stéphanie BUISINE, Jérôme GUEGAN, Frédéric SEGONDS, Fabrice MANTELET, Améziane AOUSSAT 71

Le virtuel, miroir des représentations du sujet
Virtual reality: a reflection of one's representations
Sophie MALTHET (1), Raphaële MILJKOVITCH 95

Editorial

R.I.H.M., *Revue des Interactions Humaines Médiatisées*, qualifiante en sciences de l'information et de la communication, continue à creuser le sillon de l'interdisciplinarité en croisant les méthodologies et les protocoles autour d'objets partagés. Ainsi ce nouveau numéro propose-t-il une fois encore un empan qui va de la conception des interfaces écologiques aux socialités virtuelles, en passant par l'exposé de deux méthodes originales : une méthode d'enquête qualitative pour comprendre la micro-dynamique de l'expérience d'une personne, et la mise en œuvre de la méthodes des Personas.

Le premier article nous vient à nouveau de l'Université de Sherbrooke et complète le travail initial paru dans RIHM volume 16 N° 2 en 2015 concernant les interfaces écologiques. Ce deuxième opus présente cette fois le langage WoDoMoLEID comme nouvelle technique de représentation de domaines de travail pour la conception de telles interfaces. L'article démontre comment cette technique solutionne les deux problèmes associés à la hiérarchie d'abstraction et de décomposition, d'une part, en soutenant la conception d'interfaces écologiques et, d'autre part, en établissant une correspondance directe entre les contraintes du domaine de travail et des composants visuels réutilisables.

Le deuxième article expose REMIND, une méthode d'enquête qualitative pour comprendre la micro-dynamique de l'expérience d'une personne. L'expérience y est abordée en tant que construction de sens et vécu émotionnel. Mise en œuvre et testée sur le terrain de la visite de musée, cette méthode est applicable dans d'autres contextes comme la sécurité et la santé où la construction de sens et les émotions jouent un rôle important dans la prise de décision.

Le troisième article présente deux expérimentations visant à tester l'impact des technologies collaboratives sur l'efficacité de la méthode des Personas. Il en ressort que les technologies support agissent positivement sur la collaboration et la créativité autour de la méthode des Personas, ce qui permet d'optimiser la phase d'anticipation des besoins des utilisateurs. Sont alors discutés les avantages et les inconvénients de l'usage de ces technologies dans les projets de conception.

Enfin, le dernier article considère les réseaux sociaux virtuels et les jeux en ligne massivement multijoueurs sous l'angle de l'attachement. Il examine ainsi la manière dont la psychologie peut éclairer et aider à comprendre ce qui détermine les modes d'utilisation de ces nouveaux médias. Les hypothèses posées ont été mises à l'épreuve à travers une recherche sur Facebook parmi un échantillon d'adolescents et de jeunes adultes.

Nous vous souhaitons à toutes et à tous une très bonne lecture et nous vous remercions de votre fidélité.

Sylvie **LELEU-MERVIEL** et Khaldoun **ZREIK**
Rédacteurs en chef

WoDoMoLEID : un langage de modélisation de domaines de travail pour la conception d'interfaces écologiques

WoDoMoLEID: Work Domain Modeling Language for Ecological Interface Design

Alexandre MOÏSE (1), Jean-Marc ROBERT (2)

(1) Département de systèmes d'information et méthodes quantitatives de gestion, École de gestion, Université de Sherbrooke
alexandre.moise@usherbrooke.ca

(2) Département de mathématiques et de génie industriel, École Polytechnique de Montréal
jean-marc.robert@polymtl.ca

Résumé. Une interface écologique soutient simultanément les trois niveaux de contrôle cognitif de la taxonomie SRK. Ainsi, celle-ci doit représenter les objets, les contraintes et la hiérarchie fonctionnelle d'un domaine de travail. La représentation du domaine de travail est la première étape de la conception d'interfaces écologiques. Or, la seule technique présentée à cette fin dans la littérature, la hiérarchie d'abstraction et de décomposition, est confrontée à deux problèmes. Premièrement, sa structure n'est pas compatible avec celle de l'interface écologique. Deuxièmement, pour représenter tous les éléments pertinents à la conception d'interfaces écologiques, une liste de variables et une liste d'équations doivent être ajoutées à la hiérarchie d'abstraction et de décomposition. Cet article présente le langage WoDoMoLEID comme nouvelle technique de représentation de domaines de travail pour la conception d'interfaces écologiques. L'article démontre comment cette technique solutionne les deux problèmes associés à la hiérarchie d'abstraction et de décomposition, d'une part, en soutenant la conception d'interfaces écologique et, d'autre part, en établissant une correspondance directe entre les contraintes du domaine de travail et des composants visuels réutilisables.
Mots-clés. Conception d'interfaces écologiques, analyse du domaine de travail, langage de modélisation.

Abstract. An ecological interface simultaneously supports the three levels of cognitive control of the SRK taxonomy. Thus, an ecological interface should represent objects, constraints and the functional hierarchy of a work domain. Work domain representation is the first step of ecological interface design. However, the only technique presented to this end in the literature, the abstraction-decomposition hierarchy, is facing two problems. First, its structure is not compatible with that of the ecological interface. Second, to represent every element relevant to ecological interface design, a list of variables and a list of equations must be added to the

abstraction-decomposition hierarchy. This article presents the WoDoMoLEID language as a novel work domain representation technique for ecological interface design. This article demonstrates how this technique solves the two problems associated with the abstraction-decomposition hierarchy, on the one hand, by supporting ecological interface design and, on the other hand, by establishing a direct mapping between work domain constraints and reusable visual components.
Keywords. Ecological interface design, work domain analysis, modeling language.

1 Introduction

Une interface écologique est un type particulier d'interface utilisateur permettant de soutenir simultanément les trois niveaux de contrôle cognitif de la taxonomie SRK[1] (Vicente et Rasmussen 1992). La taxonomie SRK proposée par Rasmussen (1983) postule que l'information peut être interprétée de trois manières différentes et chaque interprétation est associée à un des trois niveaux de contrôle : comportement fondé sur les habiletés (CFH), comportement fondé sur les règles (CFR) et comportement fondé sur les connaissances (CFC). Les niveaux CFH et CFR permettent de soutenir des évènements anticipés, qu'ils soient familiers ou non. Or, la particularité d'une interface écologique vient de sa capacité à soutenir le niveau CFC qui est associé aux évènements imprévus, donc qui ne peuvent être prévus.

Afin de soutenir simultanément les trois niveaux de la taxonomie SRK, Vicente et Rasmussen (1992) présentent trois principes généraux sur lesquels doit s'appuyer la conception d'interfaces écologiques :

- L'utilisateur doit pouvoir manipuler les objets du domaine de travail directement sur l'interface utilisateur (niveau CFH) ;
- Une correspondance un à un doit exister entre les contraintes du domaine de travail et les informations présentées par l'interface utilisateur (niveau CFR) ;
- l'interface utilisateur doit présenter la hiérarchie fonctionnelle du domaine de travail (niveau CFC).

Concrètement, une interface écologique doit donc représenter les objets, les contraintes et la hiérarchie fonctionnelle d'un domaine de travail. Ainsi, la conception d'interfaces écologiques débute avec l'analyse du domaine du travail (Burns et Hajdukiewicz, 2004 ; Morineau et Billet 2007). Moïse et Robert (2015) indiquent que, dans une perspective de conception d'interfaces écologiques, la littérature propose la hiérarchie d'abstraction et de décomposition comme étant la seule technique de représentation du domaine de travail. Or, puisque les premières publications sur cette technique sont apparues avant les premières publications sur les interfaces écologiques, ils viennent à la conclusion que la hiérarchie d'abstraction et de décomposition a été adoptée pour la conception d'interfaces écologiques à défaut d'avoir une meilleure technique de représentation. De plus, ils indiquent que pour concevoir une interface écologique, la hiérarchie d'abstraction et de décomposition doit être accompagnée de techniques de représentation supplémentaires comme une liste de variables et une liste d'équations. Ces constats les ont menés à réaliser une évaluation de la compatibilité entre la structure de la hiérarchie d'abstraction et de décomposition et celle de l'interface écologique en comparant leur méta-modèle respectif. Les résultats de l'évaluation confirment l'incompatibilité structurelle. Celle-ci disparaît, toutefois, en ajoutant une liste de variables et une liste d'équations à la hiérarchie d'abstraction et de décomposition. Par contre, Moïse et Robert (2015)

[1] Les initiales « SRK » proviennent de la première lettre du nom en anglais de chacun des trois niveaux de la taxonomie : ***S****kill-based behavior*, ***R****ule-based behavior*, ***K****nowlede-based behavior*.

ajoutent que l'intégration de ces trois techniques de représentation de domaine de travail mène à la description de plus d'éléments que nécessaire pour la conception d'interfaces écologiques.

Moïse et Robert (2015) soulèvent donc la pertinence de proposer de nouvelles techniques de représentation de domaine de travail dans une perspective de conception d'interfaces écologiques. Ils spécifient que ces techniques devront se conformer à deux critères. Premièrement, leur structure doit être compatible à celle de l'interface écologique. Deuxièmement, elles doivent représenter uniquement l'ensemble des éléments pertinents à la conception d'interfaces écologiques. La conformité à ces critères permet à une technique de représentation de domaines de travail, d'une part, de soutenir la conception d'interfaces écologique et, d'autre part, d'établir une correspondance directe entre les contraintes et des composants visuels réutilisables.

Outre la hiérarchie d'abstraction et de décomposition, la littérature semble très mince à ce sujet pour la conception d'interfaces écologiques. Notamment, Rechard (2015) ainsi que Rechard *et al.* (2015) proposent de représenter le domaine de travail avec un schéma synoptique et un schéma fonctionnel. Le premier schéma représente les éléments associés aux deux niveaux inférieurs de la hiérarchie d'abstraction et de décomposition et le second schéma représente les éléments des trois autres niveaux[2]. Or, les auteurs n'adressent pas les deux critères énoncés par Moïse et Robert (2015). Conséquemment, cet article a comme objectif de présenter le langage WoDoMoLEID[3]. Puisqu'il s'appuie sur la même structure que celle de l'interface écologique, ce langage est conforme aux deux critères. La section 2 présente le langage WoDoMoLEID tandis que la section 3 présente son évaluation en termes de capacité à soutenir la conception d'interfaces écologiques et d'établir une correspondance directe entre les contraintes et des composants visuels réutilisables.

2 Le langage de modélisation proposé

La présentation du langage proposé est scindée en deux parties. La première partie, présentée à la section 2.1, décrit la syntaxe abstraite (structure) et la syntaxe concrète (représentation visuelle) du langage WoDoMoLEID. La deuxième partie, présentée à la section 2.2, décrit le mécanisme d'extension du langage lui permettant d'ajouter des éléments.

2.1 Description du langage WoDoMoLEID

Kelly et Tolvanen (2008) divisent la syntaxe d'un langage de modélisation en syntaxe abstraite et syntaxe concrète. La syntaxe abstraite, normalement spécifiée avec un méta-modèle, définit les éléments du langage de modélisation et les relations entre ceux-ci, c'est-à-dire sa structure. Elle peut s'apparenter au vocabulaire et à la grammaire du langage. Quant à elle, la syntaxe concrète définit l'apparence des éléments de la syntaxe abstraite. Elle consiste en la définition d'une notation visuelle, c'est-à-dire une notation symbolique, afin de créer et manipuler des modèles. Un symbole particulier est assigné à chaque élément concret du méta-modèle.

[2] La hiérarchie d'abstraction et de décomposition contient généralement cinq niveaux qui sont, du plus élevé au plus bas : buts fonctionnels, fonctions abstraites, fonctions généralisées, fonctions physiques et forme physique (Rasmussen 1983). Toutefois, il est possible que le nombre de niveaux et leur nom puissent varier d'un domaine à l'autre (Rasmussen et Vicente 1992).

[3] Se prononce « wodomoleyd » qui est l'acronyme de : ***Wo**rk **Do**main **Mo**deling **L**anguage for **E**cological **I**nterface **D**esign*.

Afin d'assurer la meilleure compatibilité possible avec la structure d'une interface écologique, le méta-modèle du langage WoDoMoLEID s'appuie sur celui de l'interface écologique proposé par Moïse et Robert (2015). Les éléments principaux de ce nouveau langage sont les suivants : terme, opérateur et contrainte. Le méta-modèle décrit dans cette section est divisé selon ces derniers.

Le méta-modèle du langage WoDoMoLEID est représenté à l'aide du langage UML, plus spécifiquement à l'aide du diagramme de classe et, dans certains cas, du diagramme d'objets pour représenter les instances des méta-classes (Object Management Group, 2015). Moïse et Robert (2015) indiquent que « Le choix de ces langages repose sur la grande quantité de publications qui les privilégient pour la modélisation et la méta-modélisation. D'autre part, ces langages s'appuient sur le paradigme orienté objet qui est largement accepté et maîtrisé en informatique et sur lequel on s'appuie généralement pour développer des logiciels de modélisation. » C'est notamment le cas pour le prototype de logiciel de modélisation présenté à la section 3.1.1.

La figure 1 illustre les trois paquetages qui composent le méta-modèle du langage WoDoMoLEID. Le contenu de chacun de ces paquetages est détaillé dans les trois sections suivantes : la section 2.1.1 détaille le paquetage des termes, la section 2.1.2 détaille le paquetage des opérateurs (qui se divise en trois sous-paquetages) et la section 2.1.3 détaille le paquetage des contraintes. Les cinq flèches pointillées représentent les dépendances entre les paquetages. Ces dépendances concernent des relations entre des méta-classes d'un paquetage vers un autre qui sont décrites dans les trois prochaines sections.

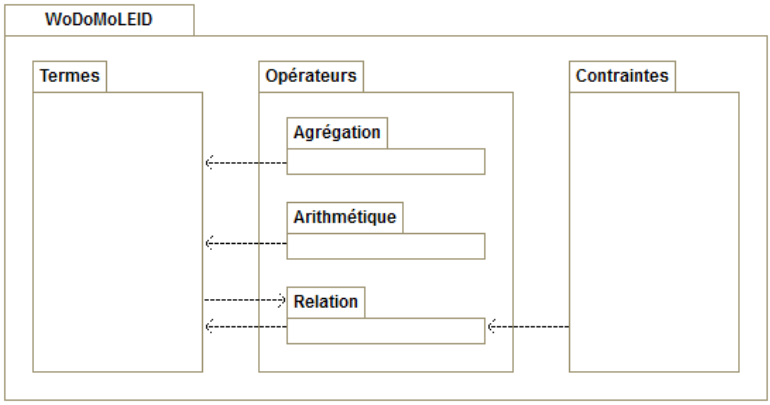


Figure 1. *Paquetages du méta-modèle*

2.1.1 Paquetage des termes

Ce paquetage contient les éléments d'un domaine de travail auxquels il est possible d'assigner une valeur. Si la valeur peut être modifiée dans le temps, il s'agit d'une variable. Dans le cas contraire, il s'agit d'une constante. Les termes doivent nécessairement être regroupés par des concepts qui représentent les objets du domaine de travail ; un concept est donc propriétaire de termes ou de concepts. La figure 2 illustre le paquetage des termes. Il s'agit de la même structure du paquetage des concepts et des termes du méta-modèle d'une interface écologique illustré à la figure 7 de Moïse et Robert (2015).

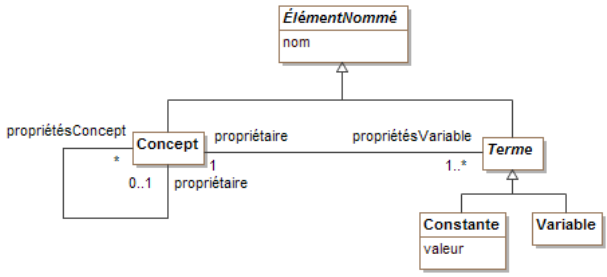


Figure 2. *Paquetage des termes*

Les méta-classes `Concept` et *`Terme`* héritent de la méta-classe abstraite *`ÉlémentNommé`*. Cette dernière définit un élément dont les instances sont déterminées par un nom. Cette généralisation évite de devoir ajouter l'attribut `nom` à différentes méta-classes, évitant ainsi une redondance.

La figure 3 illustre la syntaxe concrète de la méta-classe `Concept`. Elle a comme seul attribut `nom` qui est hérité de la méta-classe abstraite *`ÉlémentNommé`*. La valeur de ce dernier est une chaîne de caractères écrite en gras et en minuscules, sauf la première lettre qui est en majuscule. Si cette chaîne comporte plusieurs mots, ceux-ci sont concaténés sans espaces et leur première lettre est en majuscule.

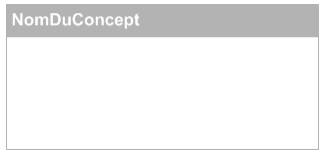


Figure 3. *Syntaxe concrète de la méta-classe* `Concept`

La figure 4 présente la syntaxe concrète des méta-classes `Variable` et `Constante` qui sont les spécialisations de la méta-classe abstraite *`Terme`*. L'affichage de la valeur de l'attribut nom qui est hérité de la méta-classe abstraite *`ÉlémentNommé`* est assujetti aux mêmes règles que la valeur de l'attribut `nom` d'une instance de la méta-classe `Concept` excepté que la première lettre du premier mot doit être une minuscule. En ce qui concerne une instance de la méta-classe `Constante`, la valeur de l'attribut `valeur` est affichée entre crochets après la valeur de l'attribut `nom`.

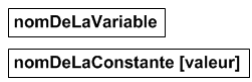


Figure 4. *Syntaxe concrète des méta-classes* `Variable` *et* `Constante`

Les concepts regroupent d'autres concepts et des termes. Cette relation est représentée par l'inclusion à l'intérieur du symbole du concept. Il s'agit de la représentation des rôles d'associations `propriétaire`, `propriétésConcept` et `propriétésTerme`.

La figure 5 présente un exemple d'inclusion. Un concept nommé « Commande » est propriétaire (rôle `propriétaire`) de deux variables (rôle

`propriétésTerme`) nommées « date » et « total » et d'un concept (rôle `propriétésConcept`) nommé « LigneCmde ». Ce dernier est propriétaire (rôle `propriétaire`) de deux variables (rôle `propriétésTerme`) nommées « quantité » et « total ». Inversement, ces éléments sont la propriété (rôle `propriétaire`) du concept qui les inclut directement. En effet, la variable « quantité » est la propriété du concept « LigneCmde » et non la propriété du concept « Commande ».

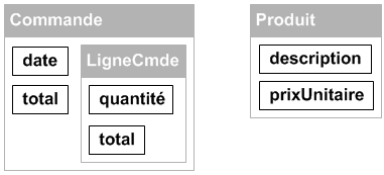


Figure 5. *Exemples d'inclusion*

2.1.2 Paquetage des opérateurs

Ce paquetage permet de créer des expressions mathématiques qui servent à définir les contraintes d'un domaine de travail. Il se démarque toutefois du paquetage des expressions du méta-modèle d'une interface écologique illustré à la figure 8 de Moïse et Robert (2015). Ce dernier permet de rassembler un nombre indéterminé d'opérateurs et de termes dans une seule expression mathématique. Ceci mène à des expressions mathématiques complexes qui peuvent être, d'une part, difficiles à valider et, d'autre part, difficiles à transformer en composants visuels graphiques d'une interface écologique. C'est pour cette raison que le paquetage des opérateurs impose des règles d'association limitant le nombre de termes et d'opérateurs d'une expression mathématique. Ces règles sont à la base de l'algorithme de détermination du niveau de chaque contrainte de la hiérarchie fonctionnelle présenté à la section 2.1.3. En somme, plutôt que de définir quelques contraintes avec des expressions mathématiques complexes, le langage WoDoMoLEID privilégie la définition de plusieurs contraintes avec des expressions mathématiques simples.

La figure 6 illustre le méta-modèle des opérateurs. La méta-classe abstraite *`Opérateur`* se spécialise en trois méta-classes abstraites (*`Agrégation`*, *`Arithmétique`* et *`Relation`*) dont chacune représente un type d'opérateur obéissant à des règles particulières d'association avec des termes ou des opérateurs. L'attribut symbole est hérité par chacune de ces trois méta-classes abstraites.

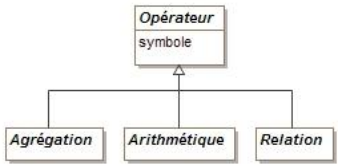


Figure 6. *Méta-modèle des opérateurs*

La syntaxe concrète d'un opérateur est illustrée à la figure 7. Il s'agit d'un cercle avec au centre le symbole de l'opérateur ; le « S » est remplacé par le symbole approprié.

Figure 7. *Syntaxe concrète de la méta-classe abstraite* `Opérateur`

Afin de gérer la complexité du paquetage des opérateurs, celui-ci est scindé en trois sous-paquetages distincts ; un pour chaque type d'opérateur comme illustré à la figure 1. Les sections suivantes décrivent chacun de ces sous-paquetages.

2.1.2.1 Agrégation

Un opérateur d'agrégation consiste à faire une opération sur une collection de valeurs d'un même terme. Le résultat de cette opération est représenté par une valeur quantitative. Le langage de modélisation proposé ne fait état que de trois opérateurs d'agrégation : `Max`, `Min` et `Somme`. Le premier consiste à obtenir la valeur maximale de la collection, le deuxième consiste à obtenir la valeur minimale de la collection et le troisième consiste à additionner toutes les valeurs de la collection. La figure 8 illustre des exemples d'application d'opérateurs d'agrégation.

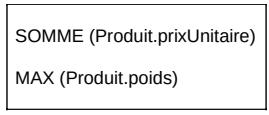


Figure 8. *Exemples d'application d'opérateurs d'agrégation*

Le paquetage des opérateurs d'agrégation est illustré à la figure 9. Une instance de la méta-classe abstraite *`Agrégation`* est obligatoirement associée à une instance de la méta-classe abstraite *`Terme`* du paquetage des termes (figure 2) qui représente l'opérande associé à l'opérateur. L'association de la méta-classe abstraite *`Agrégation`* vers la méta-classe abstraite *`Terme`* est représentée à la figure 1 par la dépendance du paquetage des opérateurs d'agrégation vers le paquetage des termes.

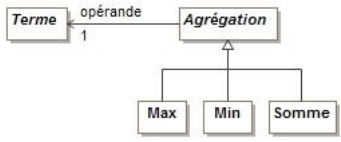


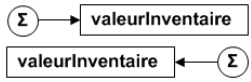
Figure 9. *Méta-modèle des opérateurs d'agrégation*

Bien que d'autres opérateurs de ce type existent, par exemple `Compte` et `Moyenne`, ils ont été omis pour des raisons de simplicité. Il est toutefois possible de les ajouter au méta-modèle par le mécanisme décrit à la section 2.2. Le tableau 1 présente les symboles de chaque type d'opérateur d'agrégation du méta-modèle de la figure 9.

Tableau 1. *Syntaxe concrète de chaque opérateur d'agrégation*

Méta-classe concrète	Symbole
Max	▲
Min	▼
Somme	Σ

La figure 10 illustre un exemple de la syntaxe concrète de la méta-classe `Somme`. La flèche avec une pointe fermée et pleine représente le rôle d'association `opérande`.



`SOMME(valeurInventaire)`

Figure 10. *Exemple de syntaxe concrète de l'opérateur d'agrégation*

Il est important de noter que selon la syntaxe abstraite, un opérateur « connaît » l'opérande auquel il est associé. Toutefois, ce dernier ne « connaît » pas l'opérateur auquel il est associé. En somme, la manière de lire les associations est de fixer en premier lieu un opérateur et ensuite d'identifier son opérande en suivant la flèche.

2.1.2.2 Arithmétique

Un opérateur arithmétique consiste à réaliser une opération sur deux variables quantitatives appelées opérandes. Le résultat de cette opération est représenté par une valeur quantitative. Pour les opérateurs arithmétiques de soustraction et de division, puisqu'ils sont non commutatifs contrairement à l'addition et à la multiplication, l'ordre des opérandes est important. Bien qu'une expression arithmétique puisse comporter plusieurs opérateurs arithmétiques, le langage WoDoMoLEID limite à un seul opérateur afin d'éviter les expressions complexes. La figure 11 illustre des exemples d'application d'opérateurs arithmétiques.

Produit.prixUnitaire x LigneCmde.quantité

Groupe.sommeDesNotes ÷ Groupe.nbÉtudiants

Cmde.sousTotal + Cmde.taxes

Figure 11. *Exemples d'application d'opérateurs arithmétiques*

Le méta-modèle des opérateurs arithmétiques est illustré à la figure 12. Une instance de la méta-classe abstraite *`Arithmétique`* est obligatoirement associée à deux instances distinctes de la méta-classe abstraite *`Terme`*, l'une représentant l'opérande de gauche et l'autre représentant l'opérande de droite. Il ne peut toutefois s'agir de la même instance de la méta-classe abstraite *`Terme`*. Les deux associations de la méta-classe abstraite *`Arithmétique`* vers la méta-classe abstraite *`Terme`* est représentée à la figure 1 par la dépendance du paquetage des opérateurs arithmétiques vers le paquetage des termes.

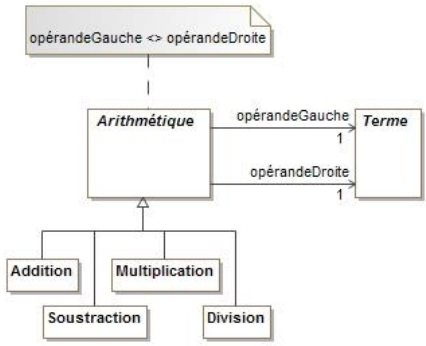


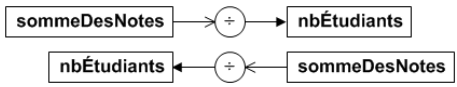
Figure 12. *Méta-modèle des opérateurs arithmétiques*

Bien que d'autres opérateurs de ce type existent, par exemple l'exposant, ils ont été omis pour des raisons de simplicité. Il est toutefois possible de les ajouter au méta-modèle par le mécanisme décrit à la section 2.2. Le tableau 2 présente le symbole de chaque type d'opérateur arithmétiques du méta-modèle de la figure 12.

Tableau 2. *Syntaxe concrète de chaque opérateur arithmétique*

Méta-classe concrète	Symbole
Addition	+
Division	÷
Multiplication	x
Soustraction	-

La figure 13 illustre un exemple de la syntaxe concrète de la méta-classe `Division`. La flèche avec la pointe ouverte représente l'association `opérandeGauche` et la flèche avec la pointe fermée et pleine représente l'association `opérandeDroite`. Ainsi, quelle que soit la position des flèches, la lecture se fait toujours de la même manière, c'est-à-dire dans l'ordre des flèches. Par conséquent, les deux représentations de la division sont équivalentes.




```
sommeDesNotes ÷ nbÉtudiants
```

Figure 13. *Exemple de syntaxe concrète de l'opérateur arithmétique*

Dans la même veine que pour l'opérateur d'agrégation, un opérateur « connaît » les opérandes auxquels il est associé. Toutefois, ces derniers ne « connaissent » pas l'opérateur auquel ils sont associés. Encore une fois, la manière de lire les associations est de fixer en premier lieu un opérateur et ensuite d'identifier ses opérandes en suivant les flèches.

2.1.2.3 Relation

Un opérateur de relation consiste à réaliser une opération sur deux opérandes. Le résultat de cette opération est représenté par une valeur booléenne, c'est-à-dire qui ne peut prendre qu'une seule de deux valeurs : vrai ou faux. Le langage WoDoMoLEID impose une instance de la méta-classe abstraite `Terme` comme opérande de gauche. En ce qui concerne l'opérande de droite, celui-ci peut être une instance de la méta-classe abstraite `Terme`, une instance de la méta-classe abstraite `Agrégation` ou une instance de la méta-classe abstraite `Arithmétique`. La figure 14 illustre des exemples d'application d'opérateurs de relation.

```
Produit.valeurInventaire = Produit.coûtUnitaire x Produit.qteEnStock

Inventaire.valeur = Σ (Produit.valeurInventaire)

Inventaire.valeurMax > Inventaire.valeur
```

Figure 14. *Exemples d'application d'opérateurs de relation*

Le paquetage des opérateurs de relation est illustré à la figure 15. Une instance de la méta-classe abstraite `Relation` est obligatoirement associée à une seule instance de la méta-classe abstraite `Terme` en tant qu'opérande de gauche. En ce qui concerne l'opérande de droite, il est associé à une méta-classe abstraite `CôtéDroit` qui peut représenter un opérateur d'agrégation, un opérateur arithmétique ou un terme. L'instance qui représente l'opérande de gauche doit être différente de l'instance qui représente l'opérande de droite. L'association de la méta-classe abstraite `Relation` vers la méta-classe abstraite `Terme` est représentée à la figure 1 par la dépendance du paquetage des opérateurs de relation vers le paquetage des termes. D'autre part, la généralisation de la méta-classe abstraite `Terme` vers la méta-classe abstraite `CôtéDroit` est représentée à la figure 1 par la dépendance du paquetage des termes vers le paquetage des opérateurs de relation.

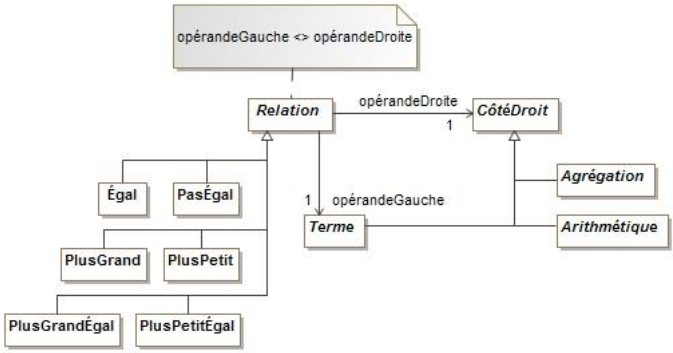


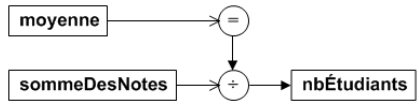
Figure 15. *Méta-modèle des opérateurs de relation*

Le tableau 3 présente les symboles de chaque type d'opérateur logique du méta-modèle de la figure 15.

Tableau 3. *Syntaxe concrète de chaque opérateur de relation*

Méta-classe concrète	Symbole
Égal	=
Pas égal	≠
Plus grand	>
Plus petit	<
Plus grand ou égal	≥
Plus petit ou égal	≤

La figure 16 illustre un exemple de la syntaxe concrète de la méta-classe `Égal`. Tout comme pour l'opérateur arithmétique, la flèche avec la pointe ouverte représente l'association `opérandeGauche` et la flèche avec la pointe fermée et pleine représente l'association `opérandeDroite`. Ainsi, quelle que soit la position des flèches, la lecture se fait toujours de la même manière, c'est-à-dire dans l'ordre des flèches.




```
moyenne = sommeDesNotes ÷ nbÉtudiants
```

Figure 16. *Exemple de syntaxe concrète de l'opérateur de relation*

La représentation de l'association entre un opérateur et ses opérandes est plus complexe pour les opérateurs de relation que pour les opérateurs arithmétiques. Dans le cas de la figure 16, l'opérande de gauche est une instance de la méta-classe `Variable`, une méta-classe qui hérite de la méta-classe abstraite *`Terme`*, et l'opérande de droite est une instance de la méta-classe abstraite *`CôtéDroit`*, plus précisément une instance de la méta-classe `Division`, une méta-classe qui hérite de la méta-classe abstraite *`Arithmétique`*.

Tout comme pour les autres types d'opérateurs, celui-ci « connaît » les opérandes auxquels il est associé, mais les opérandes ne « connaissent » pas l'opérateur auquel ils sont associés. Il faut donc, encore une fois, s'en remettre au sens des flèches. Par exemple, à la figure 16, l'opérateur de division est associé à deux variables qui sont situées à gauche et à droite et qui représentent respectivement l'opérande de gauche et l'opérande de droite. La flèche du haut n'est pas à considérer dans ce cas, car elle indique l'opérande de droite de l'opérateur d'égalité. En tant qu'opérande, l'opérateur de division n'a pas à « connaître » l'opérateur auquel il est associé, en l'occurrence l'opérateur d'égalité.

2.1.3 Paquetage des contraintes

Ce paquetage permet de définir un ensemble de contraintes dont les expressions mathématiques associées vont constituer la hiérarchie fonctionnelle. La figure 17 illustre le méta-modèle des contraintes. L'association de la méta-classe `Contrainte` vers la méta-classe abstraite *`Relation`* est représentée à la figure 1 par la dépendance du paquetage des contraintes vers le paquetage des opérateurs de relation.

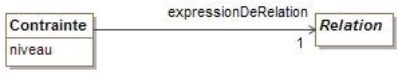


Figure 17. *Méta-modèle des contraintes*

La méta-classe `Contrainte` possède un attribut `niveau` qui identifie le niveau de la contrainte dans la hiérarchie fonctionnelle. Plus cette valeur est basse, plus elle s'approche de la description des buts du système, c'est-à-dire sa raison d'être ; une contrainte de niveau 1 est associée à un but du système. La hiérarchie fonctionnelle est obtenue par la structure des relations entre les termes utilisés à la fois comme opérande de gauche pour une contrainte et comme opérande de droite pour une autre[4].

La figure 18 présente l'algorithme permettant de déterminer le niveau de chaque contrainte. Celui-ci a été implémenté dans le prototype de logiciel de modélisation présenté à la section 3.1.1. Vicente et Rasmussen (1992) indiquent que le nombre de niveaux d'abstraction peut varier d'un domaine de travail à l'autre. Conséquemment, l'algorithme ne se limite pas aux cinq niveaux d'abstraction proposés par Rasmussen (1983).

```
Liste listeContraintes

assignerNiveaux() {
        POUR CHAQUE contrainte DE listeContraintes
                contrainte.niveau ← trouverNiveau(contrainte)
}

trouverNiveau(Contrainte contrainte) {
        niveau ← 0
        TANT QUE contrainte ≠ null
                niveau ← niveau + 1
                Terme opérandeGauche ← contrainte.opérateurDeRelation.opérandeGauche
                SI opérandeGauche ≠ null
                        contrainte ← trouverContrainte(opérandeGauche)
                SINON
                        contrainte ← null

        retourner niveau
}

trouverContrainte(Terme terme) {
        Contrainte contrainteCible ← null
        POUR CHAQUE contrainte DE listeContraintes
                CôtéDroit côtéDroit ← contrainte. opérateurDeRelation.côtéDroit
                SI côtéDroit ≠ null
                        SI côtéDroit EST Terme
                                termeDroite ← côtéDroit
                                SI termeDroite = terme
                                        contrainteCible ← contrainte
                        SINON SI côtéDroit EST Agrégation
                                opération ← côtéDroit
                                SI(opération.opérande = terme)
                                        contrainteCible ← contrainte
                        SINON SI côtéDroit EST Arithmétique
                                opération ← côtéDroit
                                SI(opération.opérandeGauche = terme OU opération.opérandeDroite = terme)
                                        contrainteCible ← contrainte

        retourner contrainteCible
}
```

Figure 18. *Algorithme permettant de déterminer le niveau de chaque contrainte*

[4] Le méta-modèle fait en sorte que, dans une hiérarchie fonctionnelle, un même terme ne peut être utilisé qu'une seule fois comme opérande de gauche d'une expression de relation et ne peut être contenu qu'une seule fois dans le côté droit d'une expression de relation.

La détermination du niveau d'abstraction d'une contrainte est réalisée en trois opérations et présuppose une liste d'objets de type `Contrainte` nommée `listeContraintes`. La première opération, appelée `assignerNiveaux()`, consiste à passer en boucle chaque élément de `listeContraintes` et assigner leur niveau en appelant l'opération `trouverNiveau()`.

L'opération `trouverNiveau()` calcule la « profondeur » d'une contrainte dans la hiérarchie fonctionnelle. Elle consiste premièrement, pour une contrainte, à sélectionner l'opérande de gauche de l'opérateur logique associé. S'il n'y en a pas, la boucle se termine et la valeur de l'attribut `niveau` est retournée. S'il existe un opérande de gauche pour cette contrainte, il faut trouver la contrainte dont l'opérateur de relation contient cet opérande, mais dans le côté droit. Ceci est réalisé par l'opération `trouverContrainte()`.

L'opération `trouverContrainte()` permet, à partir d'un terme, de trouver la contrainte dans laquelle celui-ci fait partie du côté droit de l'opérateur de relation associé à cette contrainte. Dépendamment du type de l'objet qui représente le côté droit de l'opérateur logique (*`Terme`*, *`Agrégation`* ou *`Arithmétique`*), la vérification se fait d'une manière différente.

La figure 19 présente un ensemble de contraintes, leur représentation sous forme d'un diagramme d'objets et la hiérarchie fonctionnelle sous-jacente obtenue à partir de l'algorithme de la figure 18. On remarque que la plupart des variables sont utilisées dans plus d'une contrainte. Également, la hiérarchie fonctionnelle impose des niveaux pour chaque contrainte. Par exemple, la contrainte qui porte sur le calcul des revenus et celle qui porte sur le calcul des dépenses sont au même niveau ; une n'a pas d'effet sur l'autre. Ces niveaux représentent la structure de « buts-moyens ». Par exemple, si on désire savoir *comment* faire pour atteindre l'objectif de profitabilité, on voit qu'il faut faire varier les revenus ou les dépenses. Si on désire savoir *pourquoi* on doit faire varier les revenus et les dépenses, on voit que c'est pour atteindre l'objectif de profitabilité.

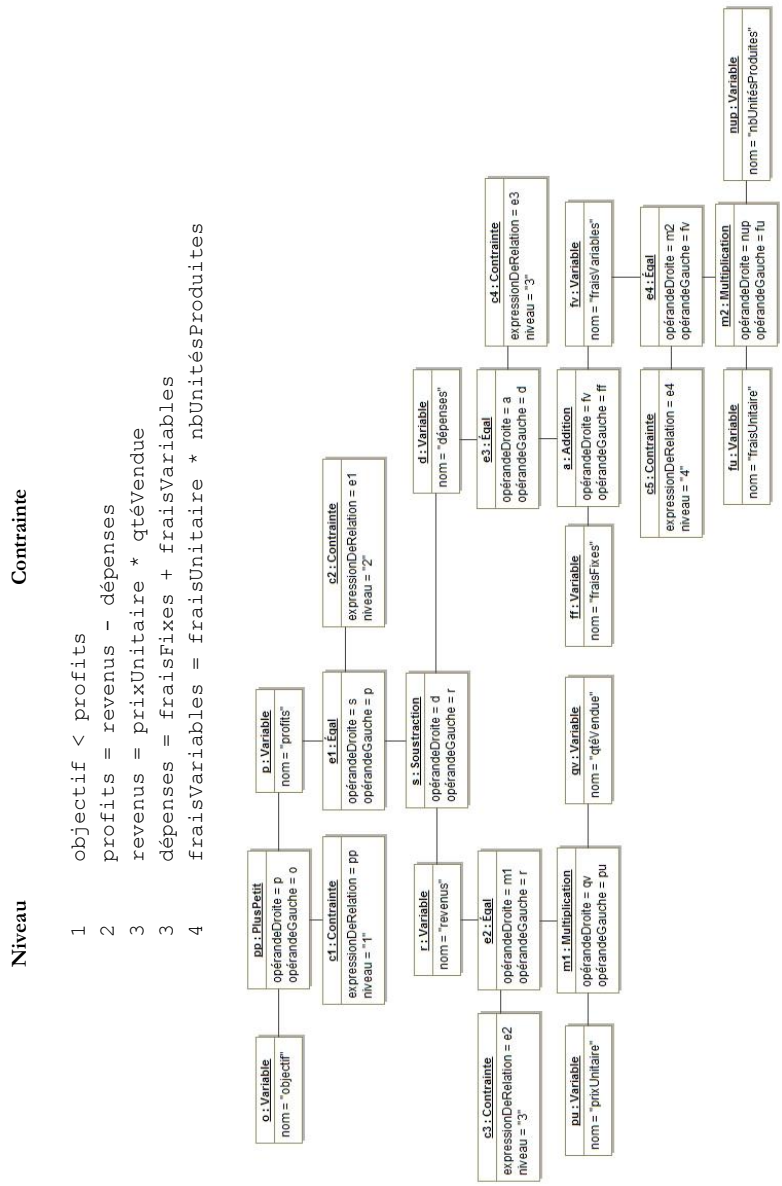


Figure 19. *Hiérarchie fonctionnelle sous-jacente à un ensemble de contraintes avec leur niveau*

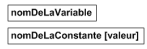
La figure 20 illustre un exemple de la syntaxe concrète de la méta-classe `Contrainte`. Une contrainte est représentée textuellement avec son niveau et l'expression de relation associée.

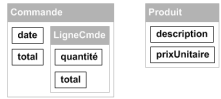
```
2  profits = revenus - dépenses
```

Figure 20. *Exemple de syntaxe concrète d'une contrainte*

Le tableau 4 présente une synthèse des principaux éléments du langage WoDoMoLEID.

Tableau 4. *Synthèse des éléments principaux du langage WoDoMoLEID*

Élément	Symbole
`Concept`	
`Terme`	

`propriétaire`
`propriétésConcept`
`propriétésTerme`
	

`Opérateur`	
`opérande` `opérandeDroite`	
`opérandeGauche`	

2.2 Extension de la syntaxe abstraite

La syntaxe abstraite du langage WoDoMoLEID est flexible en ce sens qu'il est possible d'ajouter des méta-classes. Par exemple, si nécessaire, il serait possible d'ajouter les méta-classes `Moyenne` comme spécialisations de la méta-classe abstraite *`Agrégation`* et `Exposant` comme spécialisations de la méta-classe abstraite *`Arithmétique`*. La première opération consiste à obtenir la moyenne d'une collection de valeurs. La deuxième opération consiste à obtenir le résultat de l'opérande de gauche multiplié par lui-même un certain nombre de fois tel que déterminé par l'opérande de droite. La figure 21 illustre cette situation.

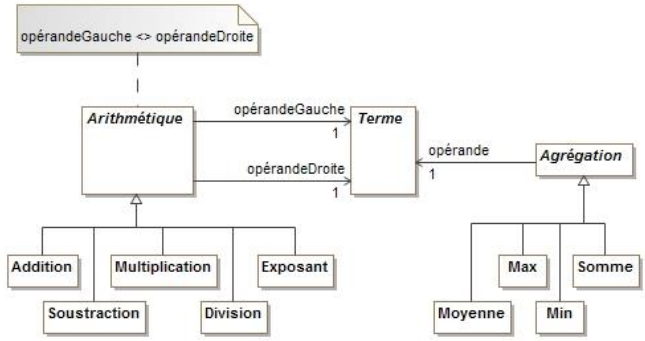


Figure 21. *Extension du méta-modèle*

Une fois positionnées dans le méta-modèle, ces nouvelles méta-classes doivent être associées à une syntaxe concrète. Le tableau 4 présente leur symbole affiché dans le cercle (syntaxe concrète de l'opérateur).

Tableau 4. *Syntaxe concrète des nouveaux opérateurs*

Méta-classe concrète	Symbole
Exposant	^
Moyenne	$\bar{x}$

3 Évaluation du langage de modélisation proposé

L'évaluation est scindée en deux parties. La première partie, présentée à la section 3.1, porte sur la capacité du langage à soutenir la conception d'interface écologique. La deuxième partie, présentée à la section 3.2, porte sur la capacité du langage à permettre une correspondance directe entre les contraintes modélisées à l'aide de ce dernier et les composants visuels qui composent une interface écologique.

3.1 Soutenir la conception d'interfaces écologiques

Tel que mentionné a la section 1, une interface écologique doit soutenir les trois niveaux de contrôle cognitif de la taxonomie SRK (Vicente et Rasmussen 1992). Pour respecter les principes généraux de conception d'interfaces écologiques, un langage de modélisation de domaines de travail doit donc permettre de décrire tous les éléments associés à chacun des trois niveaux de contrôle cognitif. Particulièrement, celui-ci doit permettre de définir les éléments suivants appartenant au domaine de travail : les objets, les contraintes et la hiérarchie fonctionnelle.

Afin d'évaluer dans ce sens le langage proposé, il est nécessaire de le mettre à l'épreuve. Pour ce faire, un prototype de logiciel de modélisation a été développé. Celui-ci implémente le méta-modèle du langage WoDoMoLEID présenté à la section 2.1 ainsi que l'algorithme de détermination du niveau de chaque contrainte présenté à la figure 18. Le logiciel permet de réaliser des modèles qui respectent les règles définies par le méta-modèle. La section 3.1.1 présente une description du prototype de logiciel de modélisation.

La section 3.1.2 présente une application du langage WoDoMoLEID réalisée avec le prototype de logiciel de modélisation à un cas fictif et simplifié de gestion de portefeuille d'actifs financiers. Ce cas d'application permet d'évaluer la capacité du langage à modéliser les éléments d'un domaine de travail pour des fins de conception d'interface écologique.

Finalement, la section 3.1.3 présente une discussion des résultats de cette première partie de l'évaluation du langage WoDoMoLEID.

3.1.1 Prototype de logiciel de modélisation

Un modèle n'a pas besoin d'être réalisé par un logiciel ; il peut être réalisé simplement en utilisant un crayon et du papier. Toutefois, un logiciel de modélisation fournit un environnement permettant, entre autres, de renforcer les règles d'un langage (Kelly et Tolvanen 2008). Par exemple, un logiciel peut indiquer si le modèle, ou un sous-ensemble de celui-ci, est incomplet et peut empêcher les associations non permises entre certains éléments. En somme, un logiciel de modélisation implémente la syntaxe abstraite et la syntaxe concrète d'un langage de modélisation.

Une autre raison pour le développement d'un prototype de logiciel de modélisation est la validation de l'algorithme présenté à la section 2.1.3 (figure 18) qui permet de déterminer le niveau de chaque contrainte. Cet algorithme est implémenté dans le logiciel et devrait permettre d'assigner automatiquement chaque contrainte à un niveau particulier.

La figure 22 présente l'interface utilisateur du logiciel de modélisation. Celui-ci comporte quatre sections. La première section, en haut à l'extrême gauche, est la barre

d'outils. Elle permet de sélectionner les fonctionnalités de création, modification et suppression des différents symboles du langage de modélisation. La deuxième section, en haut au centre, est le canevas de modélisation. C'est sur le canevas que l'utilisateur peut créer un modèle à partir des éléments de la barre d'outils. La troisième section, en bas à gauche, est celle des avertissements. Elle indique qu'une ou plusieurs règles de modélisation sont enfreintes, par exemple un terme qui n'est pas associé à un concept ou une contrainte incomplète. La quatrième section, en bas à droite, est la hiérarchie des contraintes. Elle affiche le résultat de l'algorithme permettant de déterminer le niveau de chaque contrainte afin d'obtenir la hiérarchie fonctionnelle.

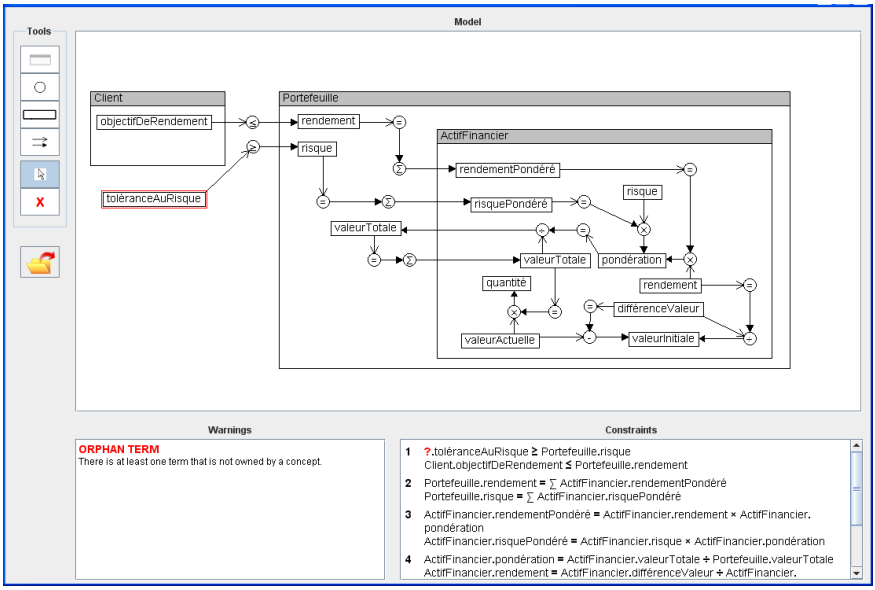


Figure 22. *Interface utilisateur du prototype de logiciel de modélisation*

3.1.2 Application à la gestion de portefeuille d'actifs financiers

Un portefeuille d'actifs financiers peut être considéré comme un système dont les objectifs sont (a) de maximiser le rendement et (b) de préserver le niveau de risque sous un certain seuil. Ces deux objectifs sont conflictuels en ce sens qu'ils imposent des restrictions l'un sur l'autre. D'une part, la maximisation du rendement est limitée par le seuil de risque ; pour un niveau de risque donné, il est peu probable d'aller au-delà d'un certain niveau de rendement. D'autre part, la réduction du risque réduit considérablement la probabilité d'atteindre le rendement maximum ; à cause de la relation risque-rendement, réduire le seuil de risque implique une réduction du rendement potentiellement atteignable. En somme, il s'agit d'optimiser le rendement en fonction du risque.

Ce système est régi par un ensemble de contraintes pouvant être représentées sous la forme d'équations mathématiques. La valeur du rendement et du risque du portefeuille est influée par la valeur actuelle, la valeur à l'achat, le risque et la quantité de chaque actif financier. Le gestionnaire de portefeuille peut contrôler uniquement la quantité de chaque actif financier par leur achat et vente.

Ce cas est entièrement quantitatif, il contient deux objectifs conflictuels, un nombre de concepts (3), de termes (15) et de contraintes (11) assez élevé pour ne pas être trop simpliste, mais assez faible pour ne pas être trop complexe. Par conséquent, il semble approprié pour la présente évaluation. La figure 23 illustre le modèle du domaine de travail pour le cas de gestion de portefeuille d'actifs financiers. Il a été réalisé à l'aide du langage WoDoMoLEID avec le prototype de logiciel à cette fin.

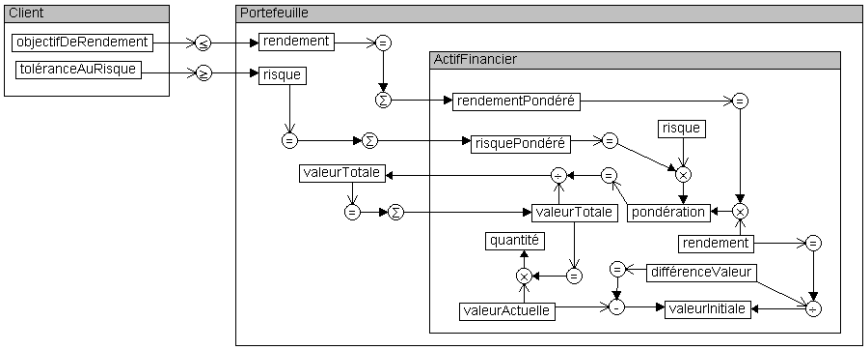


Figure 23. *Modèle du domaine de travail*

La figure 24 présente la hiérarchie fonctionnelle dérivée du modèle du domaine de travail illustré à la figure 23. Il s'agit de l'ensemble des contraintes du domaine de travail dont chacune est associée à un niveau d'abstraction particulier.

1 Client.toléranceAuRisque ≥ Portefeuille.risque
Client.objectifDeRendement ≤ Portefeuille.rendement

2 Portefeuille.rendement = ∑ ActifFinancier.rendementPondéré
Portefeuille.risque = ∑ ActifFinancier.risquePondéré

3 ActifFinancier.rendementPondéré = ActifFinancier.rendement × ActifFinancier.pondération
ActifFinancier.risquePondéré = ActifFinancier.risque × ActifFinancier.pondération

4 ActifFinancier.pondération = ActifFinancier.valeurTotale ÷ Portefeuille.valeurTotale
ActifFinancier.rendement = ActifFinancier.différenceValeur ÷ ActifFinancier.valeurInitiale

5 ActifFinancier.différenceValeur = ActifFinancier.valeurActuelle - ActifFinancier.valeurInitiale
Portefeuille.valeurTotale = ∑ ActifFinancier.valeurTotale
ActifFinancier.valeurTotale = ActifFinancier.valeurActuelle × ActifFinancier.quantité

Figure 24. *Contraintes par niveau d'abstraction*

3.1.2.1 Soutenir le CFH : représenter les objets

À la lumière des figures 22 et 23, le langage de modélisation permet de déterminer les objets du domaine de travail qui doivent être présentés par l'interface écologique. Ces objets sont représentés par les concepts suivants : client, portefeuille et actif financier. Tous les termes sont associés à un de ces concepts ; ils servent à définir ceux-ci. Par exemple, le client est défini par son objectif de rendement et sa tolérance au risque.

3.1.2.2 Soutenir le CFR : représenter les contraintes

Une des particularités d'une interface écologique par rapport à une interface utilisateur traditionnelle est le fait qu'elle met l'accent non pas sur la présentation

d'informations, mais sur les relations entre celles-ci. Dans le cas d'un domaine de travail de nature quantitative, ces relations sont définies par des opérateurs mathématiques reliés à des termes. Lorsqu'il s'agit d'un opérateur de relation, l'expression mathématique correspond à une contrainte.

Les figures 22 et 23 démontrent que le langage de modélisation permet de déterminer les contraintes d'un domaine de travail. Chaque contrainte est définie par une expression de relation pouvant prendre la valeur « vrai » ou « faux ». Tant que la valeur de chaque contrainte est « vrai », le système est dans un état normal. Si la valeur d'une contrainte est « faux », celle-ci est brisée et le système est dans un état anormal.

3.1.2.3 Soutenir le CFC : représenter la hiérarchie fonctionnelle

La hiérarchie fonctionnelle peut être définie comme étant l'ensemble interrelié de contraintes d'un domaine de travail. Il s'agit de l'ensemble de tous les termes qui sert à représenter le domaine de travail à travers différents niveaux d'abstraction.

La hiérarchie fonctionnelle peut être dérivée à l'aide de l'algorithme présenté à la section 2.1.3 (figure 18). Cet algorithme permet d'assigner un niveau d'abstraction spécifique à une contrainte en fonction de sa « profondeur » dans la hiérarchie fonctionnelle. La figure 24 présente la hiérarchie fonctionnelle qui s'appuie sur le modèle illustré à la figure 23. La valeur de chaque terme du côté droit de chaque contrainte est expliquée par une contrainte de niveau inférieur.

Les informations présentées à la figure 24 peuvent être réorganisées pour faire ressortir davantage les relations entre les termes associés à un des cinq niveaux d'abstraction. Le résultat est illustré à la figure 25 qui présente la hiérarchie fonctionnelle des termes. Chacune des relations (lignes) de cette structure est définie par un opérateur mathématique lorsque modélisée à l'aide du langage WoDoMoLEID. Il est important de garder à l'esprit que ce sont les contraintes qui sont associées aux niveaux d'abstraction et non les termes. C'est pour cette raison que les termes `ActifFinancier.valeurTotale` et `ActifFinancier.valeurInitiale` chevauchent deux niveaux d'abstraction ; ces termes sont utilisés à la fois par des contraintes de niveau 4 et 5. Quatre termes ne sont pas reliés à des termes de niveaux inférieurs : `ActifFinancier.quantité`, `ActifFinancier.valeurActuelle`, `ActifFinancier.valeurInitiale` et `ActifFinancier.risque`. La raison est que la valeur de ces termes est obtenue par l'utilisateur de l'interface écologique (`ActifFinancier.quantité`) ou par une source externe (`ActifFinancier.valeurActuelle`, `ActifFinancier.valeurInitiale` et `ActifFinancier.risque`) comme un fournisseur de données financières.

La hiérarchie fonctionnelle illustrée à la figure 25 s'apparente à la structure de la hiérarchie d'abstraction et de décomposition. Est-il possible de présenter cette hiérarchie fonctionnelle de termes dans une hiérarchie d'abstraction et de décomposition ? Si oui, c'est signe que le langage WoDoMoLEID est un équivalent à la hiérarchie d'abstraction et de décomposition pour définir la hiérarchie fonctionnelle d'un domaine de travail. Par conséquent, le langage WoDoMoLEID serait conforme à la philosophie écologique d'analyse de domaines de travail.

La figure 26 illustre la transposition de la hiérarchie fonctionnelle de la figure 25 en une hiérarchie d'abstraction et de décomposition. Chaque terme peut être associé à une case, c'est-à-dire à un niveau d'abstraction et à un niveau de décomposition, sauf pour deux termes qui sont associés à deux niveaux d'abstraction à la fois. Malgré cela, ces termes, qui ont été définis en utilisant le langage WoDoMoLEID, constituent une hiérarchie fonctionnelle compatible avec la hiérarchie d'abstraction et de

décomposition. Le langage WoDoMoLEID permet donc de représenter la même hiérarchie fonctionnelle d'un domaine de travail que si elle avait été obtenue en utilisant une hiérarchie d'abstraction et de décomposition.

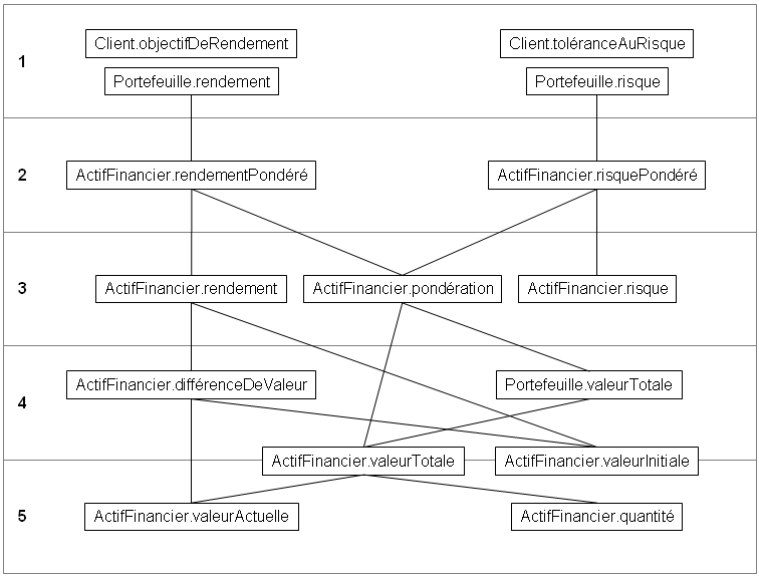


Figure 25. *Hiérarchie fonctionnelle des termes*

	Client	Portefeuille	Actif financier
1	objectifDeRendement toléranceAuRisque	rendement risque	
2			rendementPondéré risquePondéré
3			rendement pondération risque
4		valeurTotale	différenceDeValeur
4-5			valeurTotale valeurInitiale
5			valeurActuelle quantité

Figure 26. *Hiérarchie d'abstraction et de décomposition*

3.1.3 Discussion

Bien que Rasmussen (1983) propose cinq niveaux d'abstraction d'une hiérarchie d'abstraction et de décomposition, Rasmussen et Vicente (1992) indiquent que le nombre de niveaux d'abstraction et leur nom peuvent varier d'un domaine à l'autre. Ceci implique de définir les niveaux au préalable ou de les ajuster en cours d'analyse du domaine de travail.

Le langage WoDoMoLEID solutionne ce problème en permettant, à l'aide d'un algorithme à cet effet, de déterminer les niveaux d'abstraction. Plus particulièrement, au fur et à mesure que les contraintes sont définies à l'aide du langage, l'algorithme permet de les assigner à un niveau particulier. Ainsi, la hiérarchie fonctionnelle est

obtenue automatiquement sans avoir à se soucier de définir un nombre de niveaux d'abstraction et de les nommer.

Moïse et Robert (2015) ont démontré que la hiérarchie d'abstraction et de décomposition ne permet pas de modéliser un domaine de travail de manière à déterminer les éléments pertinents à la conception d'interface écologique, c'est-à-dire les objets, les contraintes et la hiérarchie fonctionnelle d'un domaine de travail. Pour ce faire, il faut l'accompagner d'une liste des variables et d'une liste des équations, cette dernière étant le document principal pour la conception d'interface écologique. En somme, la hiérarchie d'abstraction et de décomposition ne présente qu'une vue de haut niveau de la hiérarchie d'abstraction. Cette première partie de l'évaluation du langage WoDoMoLEID a démontré qu'il permet de déterminer toutes les informations nécessaires à la conception d'interface écologique, et ce, d'une manière intégrée.

3.2 Correspondance entre les contraintes et les composants visuels

La hiérarchie d'abstraction et de décomposition ne permet pas de décrire les contraintes du domaine de travail qui sont essentiels à la conception d'interface écologique (Moïse et Robert 2015). Pour pallier ce problème, un analyste de domaines de travail doit, en s'appuyant sur la hiérarchie fonctionnelle obtenue en utilisant la hiérarchie d'abstraction et de décomposition, créer une liste de variables et une liste d'équations.

Burns et Hajdukiewicz (2004) ont proposé un thésaurus visuel, c'est-à-dire un catalogue de composants visuels réutilisables pour la conception d'interfaces écologiques. Chaque composant visuel représente un type particulier de contrainte. Or, chaque composant visuel s'appuie sur une structure spécifique de contraintes. Ainsi, si les contraintes définies par l'analyste ne respectent pas cette structure, la correspondance avec les composants visuels ne peut pas être directe ; le concepteur doit redéfinir les contraintes afin qu'elles correspondent à la structure des composants visuels.

Pour résoudre ce problème, le langage WoDoMoLEID impose une structure dans la manière de définir les contraintes d'un domaine de travail qui est compatible avec la structure de composants visuels prédéfinis. Cette section consiste à évaluer la capacité du langage WoDoMoLEID à cet effet. Pour ce faire, la section 3.2.1 présente une maquette d'interface écologique conçue à partir du modèle du domaine de travail du cas de la gestion de portefeuille d'actifs financiers présenté aux figures 22 et 23. La conception de la maquette s'appuie sur un catalogue de composants visuels réutilisables présenté en annexe. Chaque composant visuel est associé à une manière de définir un type de contrainte avec le langage WoDoMoLEID. Finalement, la section 3.2.2 présente une discussion des résultats de cette deuxième partie de l'évaluation du langage WoDoMoLEID.

3.2.1 Application à la gestion de portefeuille d'actifs financiers

L'application au cas est réalisée en deux parties. La première partie, présentée à la section 3.2.1.1, porte sur la sélection des composants visuels. Il s'agit de démontrer que le langage WoDoMoLEID permet une correspondance directe entre les contraintes et les composants visuels. La deuxième partie, présentée à la section 3.2.1.2, porte sur l'intégration des composants visuels en une interface écologique. Il s'agit de démontrer que les composants sélectionnés peuvent être combinés pour constituer une interface écologique.

3.2.1.1 Sélection des composants visuels

La figure 23 présente les onze contraintes réparties à travers cinq niveaux d'abstraction du cas de la gestion de portefeuille d'actifs financiers. Les figures 26 à 36 présentent les composants visuels du catalogue en annexe pour chacune de ces contraintes.

La figure 27 illustre le composant visuel de l'opérateur `PlusGrandÉgal` avec deux termes comme opérandes de gauche et de droite.

(a) `Client.toléranceAuRisque ≥ Portefeuille.risque`

(b)

Figure 27. *(a) La contrainte 1 et (b) son composant visuel*

La figure 28 illustre le composant visuel de l'opérateur `PlusPetitÉgal` avec deux termes comme opérandes de gauche et de droite.

(a) `Client.objectifDeRendement ≤ Portefeuille.rendement`

(b)

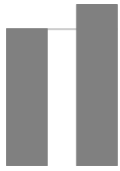


Figure 28. *(a) La contrainte 2 et (b) son composant visuel*

La figure 29 illustre le composant visuel de l'opérateur `Égal` avec un terme comme opérande de gauche et l'opérateur `Somme` comme opérande de droite.

(a) `Portefeuille.rendement = Σ ActifFinancier.rendementPondéré`

(b)

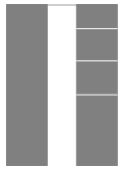


Figure 29. *(a) La contrainte 3 et (b) son composant visuel*

La figure 30 illustre le composant visuel de l'opérateur `Égal` avec un terme comme opérande de gauche et l'opérateur `Somme` comme opérande de droite.

(a) `Portefeuille.risque = Σ ActifFinancier.risquePondéré`

(b)

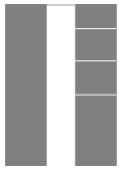


Figure 30. *(a) La contrainte 4 et (b) son composant visuel*

La figure 31 illustre le composant visuel de l'opérateur `Égal` avec un terme comme opérande de gauche et l'opérateur `Multiplication` comme opérande de droite.

(a)
```
ActifFinancier.rendementPondéré =
ActifFinancier.rendement x ActifFinancier.pondération
```

(b)

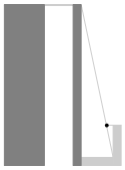


Figure 31. *(a) La contrainte 5 et (b) son composant visuel*

La figure 32 illustre le composant visuel de l'opérateur `Égal` avec un terme comme opérande de gauche et l'opérateur `Multiplication` comme opérande de droite.

(a)
```
ActifFinancier.risquePondéré =
ActifFinancier.risque x ActifFinancier.pondération
```

(b)

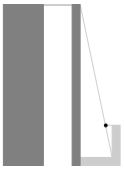


Figure 32. *(a) La contrainte 6 et (b) son composant visuel*

La figure 33 illustre le composant visuel de l'opérateur `Égal` avec un terme comme opérande de gauche et l'opérateur `Division` comme opérande de droite.

(a)
```
ActifFinancier.pondération =
ActifFinancier.valeurTotale ÷ Portefeuille.valeurTotale
```

(b)

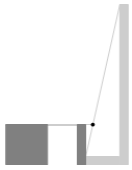


Figure 33. *(a) La contrainte 7 et (b) son composant visuel*

La figure 34 illustre le composant visuel de l'opérateur `Égal` avec un terme comme opérande de gauche et l'opérateur `Division` comme opérande de droite.

(a)
```
ActifFinancier.rendement =
ActifFinancier.différenceValeur ÷ ActifFinancier.valeurInitiale
```

(b)

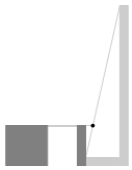


Figure 34. *(a) La contrainte 8 et (b) son composant visuel*

La figure 35 illustre le composant visuel de l'opérateur `Égal` avec un terme comme opérande de gauche et l'opérateur `Soustraction` comme opérande de droite.

(a)
```
ActifFinancier.différenceValeur =
ActifFinancier.valeurActuelle - ActifFinancier.valeurInitiale
```

(b)

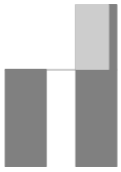


Figure 35. *(a) La contrainte 9 et (b) son composant visuel*

La figure 36 illustre le composant visuel de l'opérateur `Égal` avec un terme comme opérande de gauche et l'opérateur `Somme` comme opérande de droite.

(a)
```
Portefeuille.valeurTotale = Σ ActifFinancier.valeurTotale
```

(b)

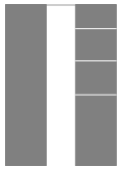


Figure 36. *(a) La contrainte 10 et (b) son composant visuel*

La figure 37 illustre le composant visuel de l'opérateur `Égal` avec un terme comme opérande de gauche et l'opérateur `Multiplication` comme opérande de droite.

(a)

```
ActifFinancier. valeurTotale =
ActifFinancier.valeurActuelle x ActifFinancier.quantité
```

(b)

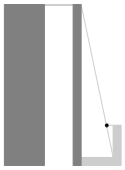


Figure 37. *(a) La contrainte 11 et (b) son composant visuel*

3.2.1.2 Intégration des composants visuels en une maquette d'interface écologique

La figure 38 illustre une maquette d'interface écologique pour le cas de gestion de portefeuille d'actifs financiers qui intègre les composants visuels sélectionnés du catalogue en annexe. Il est à noter que les composants visuels ne représentent pas de valeurs cohérentes ; il s'agit uniquement d'illustrer leur intégration en une interface écologique en les disposant sur une surface d'affichage.

Cette interface écologique présente un portefeuille contenant trois actifs financiers (AF1, AF2 et AF3). Les mêmes composants visuels sont réutilisés pour représenter les contraintes associées à chaque actif financier. Les composants visuels sont disposés sur cinq rangées. Chaque rangée correspond à un des cinq niveaux d'abstraction déterminé à la figure 23 ; les composants visuels de la première rangée du haut représentent les contraintes du premier niveau d'abstraction et les composants visuels de la dernière rangée du bas représentent les contraintes du cinquième niveau d'abstraction.

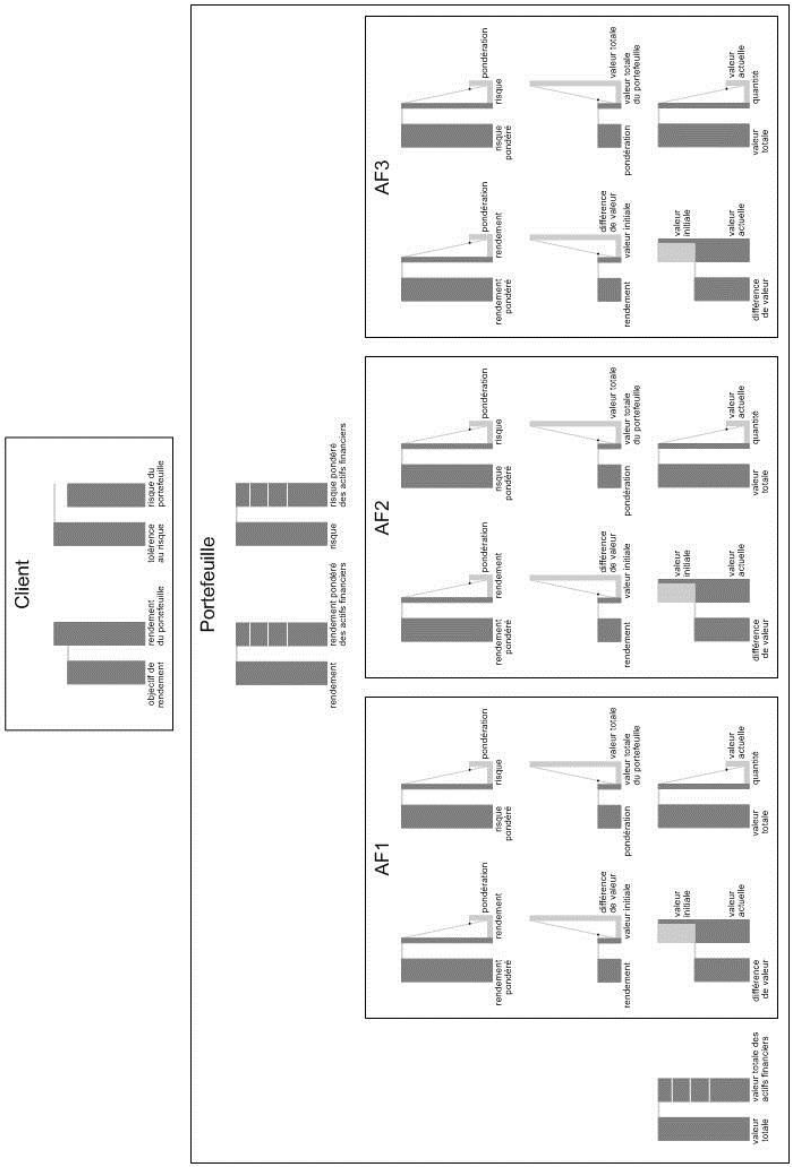


Figure 38. *Maquette d'une interface écologique pour la gestion de portefeuille d'actifs financiers*

3.2.2 Discussion

À l'annexe, un catalogue de composants visuels réutilisables est proposé. La structure de chaque composant visuel correspond à la structure d'un opérateur particulier du langage WoDoMoLEID. Ainsi, lorsque le domaine de travail est modélisé à l'aide du langage WoDoMoLEID, il est possible d'associer directement un composant visuel particulier à un type d'opérateur, par exemple une addition ou une égalité.

Cette affirmation est démontrée à la section 3.2.1.2 dans laquelle des composants visuels ont été sélectionnés afin de créer une interface écologique pour représenter un portefeuille d'actifs financiers. Cette application démontre la

réutilisation des composants visuels. En effet, le cas comporte onze contraintes réparties sur cinq niveaux d'abstraction. Plusieurs de ces contraintes s'appuient sur les mêmes opérateurs, par exemple les contraintes 3, 4 et 10 avec un opérateur `Égal` et un opérateur `Somme` ainsi que les contraintes 7 et 8 avec un opérateur `Égal` et un opérateur `Division`. Même si elles s'appuient sur des termes différents, ces contraintes sont associées aux mêmes composants visuels puisqu'elles utilisent les mêmes opérateurs. La raison est que la définition des contraintes du domaine de travail avec le langage WoDoMoLEID s'appuie sur la même structure que celle sous-jacente aux composants visuels.

Cette section a démontré que WoDoMoLEID permet une correspondance directe entre la définition des contraintes d'un domaine de travail et les composants visuels. Établir cette correspondance est le premier pas vers l'automatisation de la démarche de sélection des composants visuels. Toutefois, le travail de conception ne se limite pas à cette tâche. Le concepteur doit principalement s'acquitter de deux autres tâches : (a) la création des composants visuels et (b) la disposition de ceux-ci sur la surface d'affichage. Ces deux tâches requièrent l'application de règles ergonomiques de présentation d'informations[5]. Ces règles ne sont pas propres aux interfaces écologiques, mais aux interfaces utilisateurs en général. Puisque cet article porte exclusivement sur la conception d'interface écologique, les considérations externes à ce type d'interface utilisateur sont omises.

4 Conclusion

Cet article présente un nouveau langage de modélisation de domaines de travail pour la conception d'interfaces écologiques appelé WoDoMoLEID. Il peut être considéré comme une alternative de la hiérarchie d'abstraction et de décomposition ou il peut être utilisé conjointement avec cette technique de représentation de domaines de travail par un analyste. L'objectif de ce nouveau langage est de faciliter le travail du concepteur d'interface écologique en permettant une correspondance directe entre la définition des contraintes du domaine de travail et les composants visuels qui les représentent et qui composent l'interface écologique.

4.1 Contributions

Cet article présente trois contributions. La première et principale contribution est un nouveau langage de modélisation de domaines de travail, appelé WoDoMoLEID. Ce langage, de par sa structure, apporte une solution aux deux problèmes de la hiérarchie d'abstraction et de décomposition. Premièrement, il tient compte, de manière intégrée, de tous les éléments d'un domaine de travail nécessaire à la conception d'interfaces écologiques : objets, contraintes, hiérarchie fonctionnelle. Un symbole différent est associé à chaque élément du méta-modèle éliminant ainsi toute ambigüité dans l'interprétation de la syntaxe concrète. Deuxièmement, le langage proposé impose une structure dans la manière de définir les contraintes. Cette structure, qui dicte la manière dont les opérateurs et les termes sont reliés, permet une correspondance directe entre les contraintes et les composants visuels qui composent une interface écologique. Cette correspondance directe est un premier pas dans l'automatisation de la sélection de composants visuels réutilisables.

La deuxième contribution est un prototype de logiciel de modélisation. Ce dernier implémente la syntaxe abstraite et la syntaxe concrète du langage WoDoMoLEID. Ainsi, ce logiciel de modélisation renforce les règles du langage

[5] Pour la littérature à ce sujet, voir entre autres Nogier et Leclerc (2016), Huart *et al.* (2008), Tullis (1997), Sanders et McCormick (1993) et Mayhew (1992).

imposées par sa syntaxe abstraite. Il permet d'indiquer à son utilisateur qu'un modèle est bien formé ou non. Également, à partir d'un modèle du domaine, un algorithme permet de dériver automatiquement la hiérarchie fonctionnelle en associant un niveau de « profondeur » à chaque contrainte. L'utilisateur peut ainsi se concentrer sur la définition des contraintes du domaine de travail sans avoir à se soucier de les positionner dans la hiérarchie fonctionnelle.

La troisième contribution est une première ébauche d'un catalogue de composants visuels réutilisables présenté en annexe. Chaque composant visuel correspond à la représentation graphique d'un type d'opérateur et les relations avec ses opérandes. Ce catalogue permet de réduire les délais de conception d'interfaces écologiques en évitant de devoir créer les composants visuels à chaque nouvelle interface écologique ; le concepteur peut les sélectionner et les disposer sur la surface d'affichage.

4.2 Limites

Bien que cet article présente plusieurs contributions, il présente également certaines limites. La première limite est le degré de complexité du cas employé à la section 3 pour l'évaluation du langage, la gestion de portefeuille d'actifs financiers. Ce cas est représentatif de la réalité en ce sens que les contraintes définies sont celles réellement utilisées dans ce domaine. Néanmoins, il est raisonnable de se questionner sur la modélisation d'un domaine de travail comprenant un plus grand nombre de contraintes. Est-ce que la syntaxe concrète telle que définie à la section 2 crée un obstacle à la lisibilité du modèle pour un domaine plus complexe, c'est-à-dire qui comporte plus de concepts, de termes et d'opérateurs que le cas de la gestion de portefeuille d'actifs financiers ? Rien dans cet article ne permet de répondre à cette question.

La deuxième limite est que le langage de modélisation proposé permet de représenter des domaines de travail de nature quantitative seulement ; il ne tient pas compte de la modélisation de domaines de travail qualitatifs. Burns et Hajdukiewicz (2004) ainsi que Rasmussen, Pjetersen et Goodstein (1994) présentent quelques domaines de travail de nature qualitative. Il est impossible de les modéliser avec le langage WoDoMoLEID dans son état actuel.

La troisième limite porte sur le catalogue de composants visuels présentés en annexe. Ce dernier a été conçu uniquement pour valider la correspondance entre les contraintes définies avec le langage WoDoMoLEID et des composants visuels réutilisables. Les composants visuels ont été créés arbitrairement ; ils n'ont pas fait l'objet d'évaluations ergonomiques pur savoir s'ils sont compréhensibles par un utilisateur.

La quatrième limite est que la correspondance directe entre la structure des contraintes telles que définies par le langage WoDoMoLEID et la structure des composants visuels présentées en annexe n'est qu'un premier pas vers l'automatisation du processus de sélection des composants visuels. Toutefois, cette automatisation n'a pas été réalisée.

4.3 Avenues futures de recherche

Quelques avenues de recherches se dessinent toutefois dont certaines sont associées aux limites présentées précédemment.

La première avenue de recherche porte sur l'amélioration du langage WoDoMoLEID. Cette avenue se divise en trois voies. La première voie consiste à modifier la syntaxe abstraite de manière à ajouter des mécanismes de représentation de contraintes qualitatives. La deuxième voie consiste à modifier la syntaxe concrète pour la rendre plus utilisable. En effet, la syntaxe concrète présentée à la section 2 a

été créée arbitrairement sans savoir si les symboles utilisés pouvaient être interprétés correctement. Elle pourrait donc faire l'objet d'études expérimentales avec des sujets humains. La troisième voie porte la modélisation de domaines très complexes, c'est-à-dire qui comportent un grand nombre de concepts, de termes et d'opérateurs. Afin de permettre la lisibilité des modèles, il serait nécessaire de permettre un mécanisme visuel permettant de scinder le modèle en plusieurs sous-modèles. Une solution serait d'introduire la notion de paquetages utilisée par le langage UML (Object Management Group 2015).

La deuxième avenue de recherche, qui se divise également en trois voies, porte sur les composants visuels. La première voie consiste à enrichir le catalogue de composants visuels réutilisables. Pour ce faire, il faudrait proposer différents agencements de formes géométriques et les soumettre à des évaluations par des sujets humains. Précisément dans cette optique, Jessa et Burns (2007) ont réalisé plusieurs expériences visant à déterminer la sensibilité visuelle d'affichages graphiques dynamiques. En s'inspirant de leurs travaux, il serait intéressant d'évaluer l'impact de caractéristiques, par exemple la taille, la couleur et la disposition, sur la performance humaine. La deuxième voie porte sur l'automatisation du processus de sélection des composants visuels. Cet article présente les règles nécessaires à cet effet. Un prototype de logiciel pourrait être développé permettant de sélectionner les composants visuels appropriés à partir des contraintes définies avec le langage WoDoMoLEID. Cette voie va de pair avec la troisième qui consiste à intégrer les règles ergonomiques nécessaires à l'optimisation de l'utilisabilité de l'interface écologique comme l'ont fait Rechard (2015) ainsi que Rechard *et al.* (2015). Ces règles qui serviraient à définir, entre autres, la disposition des composants visuels sur la surface d'affichage et les mécanismes de modification de valeurs pourraient être définies et ainsi être prises en considération pour l'automatisation du processus de conception de l'interface écologique.

La troisième avenue de recherche porte sur les logiciels de modélisation. Le prototype présenté à la section 3.1.1 a été créé dans le but de mettre le langage WoDoMoLEID en application. Les sections ont été créées arbitrairement et seules les fonctionnalités de base (ouvrir, sauvegarder, manipuler et valider un modèle) ont été implémentées. Un nouveau prototype pourrait être développé en offrant d'autres fonctionnalités comme la possibilité de voir le modèle sous forme d'arborescence ou de visionner un sous-ensemble du modèle. Surtout, il est important que le prototype de logiciel de modélisation soit conçu de manière ergonomique.

Vicente (2002) présente un certain nombre de facteurs qu'il considère comme faisant obstacle à l'utilisation d'interface écologique dans l'industrie. Un de ceux-ci est associé à la complexité et la lourdeur de l'approche de conception. Ces avenues de recherche permettront, espérons-le, d'alléger ce processus en automatisant la plupart des étapes entre l'analyse du domaine de travail et l'interface écologique finale sous forme de code source. Ainsi, il sera plus facile de concevoir et réaliser des environnements de travail qui soutiendront davantage les trois niveaux de contrôle cognitif, particulièrement le niveau CFC qui est associé aux événements imprévus.

5 Bibliographie

Burns, C. M., Hajdukiewicz, J. R. (2004). *Ecological Interface Design*. CRC Press, Boca Raton.

Huart, J., Kolski, C., Bastien, C. (2008). L'évaluation de documents multimédias, état de l'art. In *Objectiver l'humain ? Volume 1, Qualification, quantification*, Leleu-Merviel, S. (Ed.), Hermes, Paris, 211-250.

Jessa, M., Burns, C. M. (2007). Visual sensitivities of dynamic graphical displays. *International Journal of Human-Computer Studies*, vol. 65, num. 3, 205-222.

Kelly, S., Tolvanen, J.-P. (2008). *Domain-specific modeling: enabling full code generation*. John Wiley & Sons, Hoboken.

Mayhew, D. J. (1992). Screen Layout and Design. In *Principles and guidelines in software user interface design*, Mayhew, D. J. (Éd.), Prentice-Hall, Englewood Cliffs, 458-506.

Moïse, A., Robert, J.-M. (2015). Évaluation de la compatibilité structurelle entre la hiérarchie d'abstraction et de décomposition et l'interface écologique. *Revue d'Interaction Homme-Machine*, vol. 15, num. 2, 3-33.

Morineau, T., Billet, H. (2007). L'analyse du domaine de travail et les cartes cognitives pour évaluer une IHM - Application à un Logiciel de Finances. *Revue d'Interaction Homme-Machine*, vol. 8, num. 2, 99-116.

Nogier, J.-F., Leclerc, J. (2016). *UX Design et ergonomie des interfaces, 6e édition*. Dunod, Malakoff.

Object Management Group (2015). *OMG Unified Modeling Language (OMG UML), version 2.5*. formal/2015-03-01.

Rasmussen, J. (1983). Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. smc-13, num. 3, 257-266.

Rasmussen, J., Pejtersen, A. M., Goodstein, L. P. (1994). *Cognitive Systems Engineering*. John Wiley & Sons, New York.

Rechard, J. (2015). *Introduction de critères ergonomiques dans un système de génération automatique d'interfaces de supervision*. Thèse de l'Université de Bretagne-Sud, Lab-STICC, novembre.

Rechard, J., Morineau, T., Léger, F., Bignon, A., Kesraoui, D., Bouillon, J.-F., Berruet, P. (2015). Experimental verification of ecological interface prototype issued by an automated generation tool. In *6th International Conference on Applied Human Factors and Ergonomics (AHFE 2015) and the Affiliated Conferences*, Ahram, T., Karwowski, W., Schmorrow, D. (Eds.), Elsevier, vol. 3, 5747-5754.

Sanders, M. S., Mccormick, E. J. (1993). Visual displays of dynamic information. In *Human Factors in Engineering and Design, seventh edition*, Sanders, M. S., McCormick, E. J. (Eds.), McGraw-Hill, New York, 132-159.

Tullis, T. S. (1997). Screen design. In *Handbook of Human-Computer Interaction, second edition*, Helander, M. G., Landauer, T. K., Prabhu, P. V. (Éds.), North Holland, Amsterdam, 503-531.

Vicente, K. J. (2002). Ecological interface design: Progress and challenges. *Human Factors*, vol. 44, num. 1, 62-78.

Vicente, K. J., Rasmussen, J. (1992). Ecological Interface Design: Theoretical Foundations. *IEEE Transactions on Systems, Man, And Cybernetics*, vol. 22, num. 4, 589-606.

6 Annexe : catalogue de composants visuels

Cette annexe présente un catalogue de composants visuels réutilisables pour la conception d'interfaces écologiques. Un composant visuel est un agencement de formes géométriques qui représente une contrainte, c'est-à-dire des relations mathématiques entre plusieurs termes. Les types de relations présentés dans cette section correspondent aux opérateurs définis à la section 2.1.2.

Le catalogue présenté à cette annexe n'utilise pas intégralement le thésaurus visuel de Burns et Hajdukiewicz (2004), mais s'en inspire grandement. Ce dernier présente plusieurs manières de représenter graphiquement différentes relations entre des termes. Le présent catalogue se limite qu'à une seule manière par type de relation, c'est-à-dire par opérateur. Pour chaque opérateur, son composant visuel associé, sa représentation avec le langage WoDoMoLEID et l'extrait du méta-modèle correspondant à cette dernière sont présentés.

Il est à noter que les composants visuels présentés dans cette annexe peuvent être modifiés ou remplacés. L'important est que leur structure corresponde à celle imposée par le langage WoDoMoLEID. Le choix de leur forme est inspiré de Burns et Hajdukiewicz (2004), mais n'a pas fait l'objet de validation auprès de sujets humains.

6.1 Agrégation

Les trois opérateurs d'agrégation décrits à la section 2.1.2.1 sont : `Max`, `Min` et `Somme`. Leur composant visuel est illustré respectivement aux figures 38, 39 et 40. Un opérateur d'agrégation consiste à réaliser une opération sur un ensemble de valeurs qui correspondent à un même terme, mais pour des objets différents. Un terme est représenté par une barre verticale grise où la valeur du terme est associée à sa hauteur.

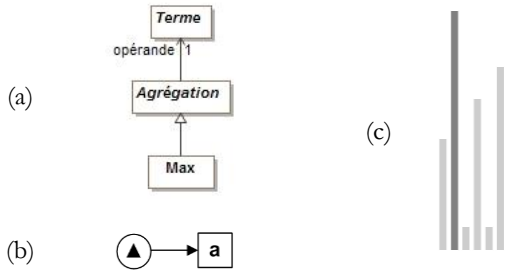


Figure 39. *(a) Extrait du méta-modèle, (b) représentation avec le langage WoDoMoLEID et (c) un exemple du composant visuel de l'opérateur* `Max`

En ce qui concerne les opérateurs `Max` et `Min`, le résultat de ces opérations consiste à identifier la valeur maximale ou minimale d'un ensemble de valeurs correspondant au terme `a`. Il est donc nécessaire de présenter chaque valeur de l'ensemble. Le résultat est déterminé à l'aide d'une nuance différente, c'est-à-dire un gris plus foncé. À la figure 39, la barre grise foncée identifie la valeur maximale de l'ensemble de six valeurs. À la figure 40, sur les six valeurs, deux correspondent à la valeur minimale de l'ensemble. Elles sont toutes deux identifiées à l'aide d'un gris plus foncé. Si les valeurs de l'ensemble sont modifiées, la valeur maximale ou minimale est toujours identifiée en foncé.

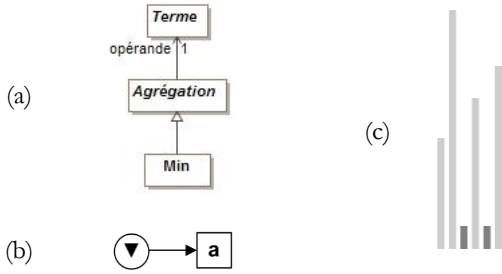


Figure 40. (*a) Extrait du méta-modèle, (b) représentation avec le langage WoDoMoLEID et (c) un exemple du composant visuel de l'opérateur* `Min`

La figure 41 illustre le composant visuel de l'opérateur `Somme`. Le résultat correspond à l'addition de toutes les valeurs de l'ensemble. Chaque valeur correspond à une section de la barre. L'exemple du composant visuel illustre le résultat de l'addition de quatre valeurs du terme `a`. Ainsi, si les valeurs de l'ensemble sont respectivement 1, 4, 3 et 2, le résultat est 10. La hauteur de chaque section de la barre doit respecter le ratio de sa valeur par rapport au résultat de la somme. Si une des valeurs de l'ensemble est modifiée, la taille de la barre est modifiée en conséquence.

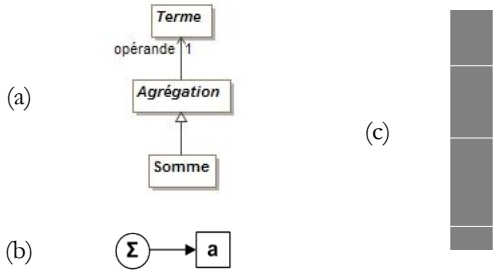


Figure 41. *(a) Extrait du méta-modèle, (b) représentation avec le langage WoDoMoLEID et (c) un exemple du composant visuel de l'opérateur* `Somme`

6.2 Arithmétique

Les quatre opérateurs arithmétiques décrits à la section 2.1.2.2 sont : `Addition`, `Soustraction`, `Multiplication` et `Division`.

La figure 42 illustre le composant visuel de l'opérateur `Addition`. Il s'agit de la combinaison de la valeur de deux termes. Le composant visuel est une barre à deux sections ; chaque section représente la valeur d'un des deux termes. Si la valeur d'un des deux termes est modifiée, la taille de la barre est modifiée en conséquence.

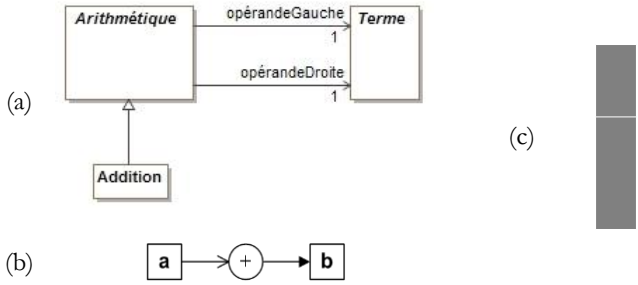


Figure 42. *(a) Extrait du méta-modèle, (b) représentation avec le langage WoDoMoLEID et (c) un exemple du composant visuel de l'opérateur* `Addition`

La figure 43 illustre le composant visuel de l'opérateur `Soustraction`. Cet opérateur porte également sur la relation entre deux termes. Deux exemples du composant visuel sont illustrés à la figure 43. L'exemple de gauche illustre la situation lorsque la valeur du terme `a` est supérieure à la valeur du terme `b`. La barre grise foncée représente la valeur du terme `a` et la barre grise pâle représente la valeur du terme `b`. Le résultat de l'opération est représenté par la section foncée inférieure. Il faut lire cette barre comme étant une barre grise foncée d'une certaine hauteur (valeur du terme `a`) à laquelle on retranche une barre grise pâle d'une certaine hauteur (valeur du terme `b`).

L'exemple de droite illustre la situation inverse, c'est-à-dire que la valeur du terme `a` est inférieure à la valeur du terme `b`. La barre grise foncée représente la valeur du terme `a` et la barre grise pâle représente la valeur du terme `b`. La section pâle inférieure représente le résultat de l'opération. Toutefois, puisque la section pâle est plus grande que la section foncée, il faut interpréter le résultat de l'opération comme étant négatif.

Dans le cas où les valeurs respectives des termes `a` et `b` sont égale, la barre aura une seule section grise foncée.

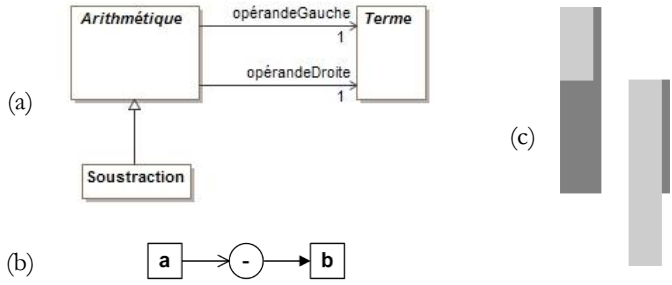


Figure 43. *(a) Extrait du méta-modèle, (b) représentation avec le langage WoDoMoLEID et (c) deux exemples du composant visuel de l'opérateur* `Soustraction`

La figure 44 illustre le composant visuel de l'opérateur `Multiplication`. Tout comme les opérations d'addition et de soustraction, la multiplication met en relation deux termes. Sa représentation géométrique est toutefois plus complexe. Deux exemples sont illustrés à la figure 44. Il s'agit d'une combinaison de trois barres. Les deux barres pâles représentent les valeurs des termes `a` (barre verticale) et `b` (barre

horizontale). La barre foncée représente le résultat de l'opération. Les trois barres sont reliées par deux lignes, une horizontale et une diagonale. La taille de la diagonale peut changer, mais celle de la ligne horizontale reste fixe. L'intersection entre ces deux lignes est représentée par un point noir. Ce point est un pivot qui représente le centre de la relation de multiplication. Par exemple, si la valeur du terme `a` augmente, la ligne horizontale et le point prendront une position plus élevée. Ceci aura pour effet de déplacer la diagonale de telle manière à positionner son extrémité supérieure gauche plus haut entraînant avec elle une augmentation de la taille de la barre foncée. Les deux exemples du composant visuels de la figure 44 illustrent un tel changement.

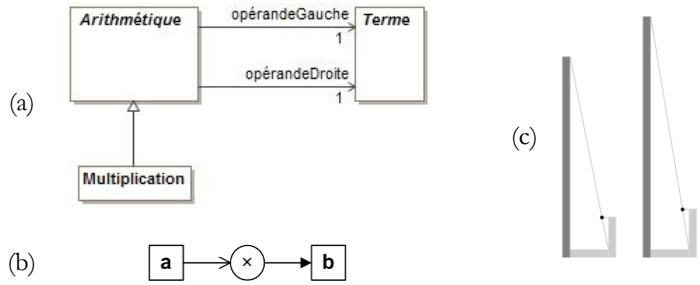


Figure 44. *(a) Extrait du méta-modèle, (b) représentation avec le langage WoDoMoLEID et (c) deux exemples du composant visuel de l'opérateur* `Multiplication`

La figure 45 illustre le composant visuel de l'opérateur `Division`. Ce composant visuel doit être interprété de la même manière que pour le composant visuel de l'opérateur `Multiplication`. Encore une fois, les barres pâles représentent les valeurs des termes `a` (barre verticale) et `b` (barre horizontale) et la barre foncée représente le résultat de l'opération.

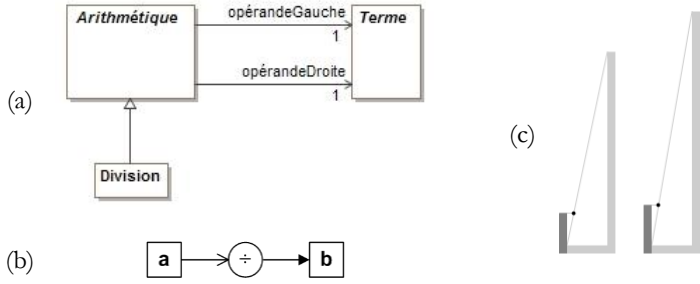


Figure 45. *(a) Extrait du méta-modèle, (b) représentation avec le langage WoDoMoLEID et (c) deux exemples du composant visuel de l'opérateur* `Division`

6.3 Relation

Les six opérateurs de relation décrits à la section 2.1.2.3 sont : `Égal`, `PasÉgal`, `PlusGrand`, `PlusGrandÉgal`, `PlusPetit` et `PlusPetitÉgal`.

La figure 46 illustre le composant visuel de l'opérateur `Égal`. Trois exemples du composant visuel sont illustrés avec des termes pour l'opérande de gauche (barre de gauche) et l'opérande de droite (barre de droite). La barre de gauche représente la valeur du terme `a` qui fait office de référence. Une ligne horizontale permet de

comparer la valeur du terme `b`, la barre de droite, à cette référence. L'exemple du centre illustre la situation lorsque le résultat de l'opération est « vrai », c'est-à-dire lorsque les valeurs des termes `a` et `b` sont égales. Les deux autres exemples illustrent la situation lorsque le résultat de l'opération est « faux ». La couleur noire est utilisée pour représenter soit le manque à gagner, soit l'excédent.

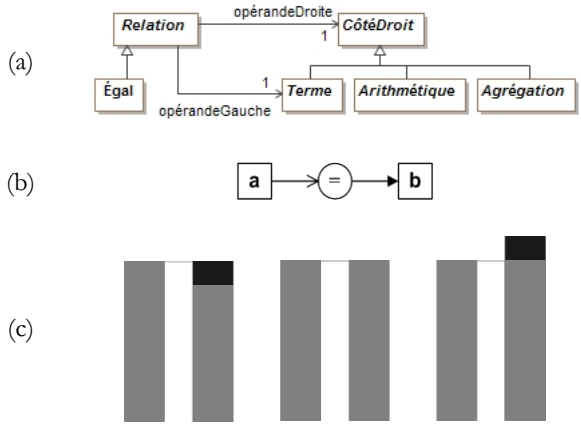


Figure 46. *(a) Extrait du méta-modèle, (b) représentation avec le langage WoDoMoLEID et (c) trois exemples du composant visuel de l'opérateur* `Égal`

La figure 47 illustre le composant visuel de l'opérateur `PasÉgal`. Seul l'exemple du centre présente un résultat d'opération « faux ». Il s'agit de la situation où les valeurs des termes `a` et `b` sont égales ; il y a donc un léger manque à gagner ou excédent représenté par la ligne noire en haut de la barre de droite. En ce qui concerne les deux autres exemples, ils représentent deux situations d'inégalités entre les valeurs des termes `a` et `b` ; l'absence de couleur noire indique que le résultat de l'opération est « vrai ».

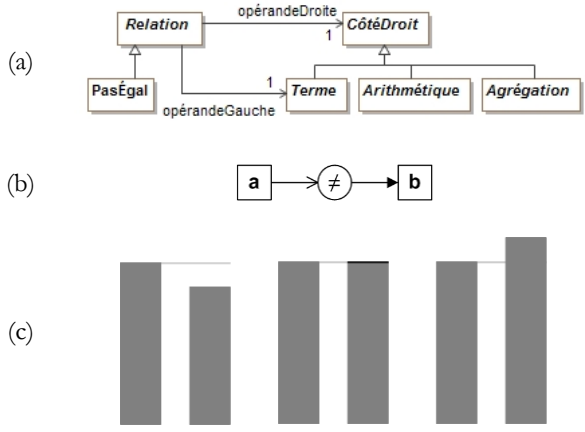


Figure 47. *(a) Extrait du méta-modèle, (b) représentation avec le langage WoDoMoLEID et (c) trois exemples du composant visuel de l'opérateur* `PasÉgal`

La figure 48 illustre le composant visuel de l'opérateur `PlusGrand`. Seul l'exemple de gauche indique un résultat « vrai » de l'opération. La couleur noire dans les deux autres exemples indique que la valeur du terme `a` n'est pas supérieure à la valeur du terme `b`.

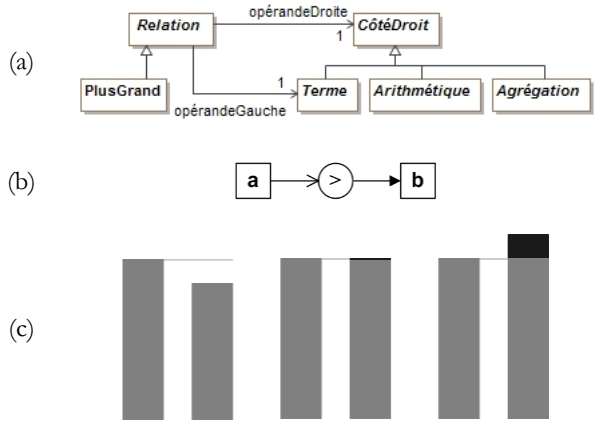


Figure 47. *(a) Extrait du méta-modèle, (b) représentation avec le langage WoDoMoLEID et (c) trois exemples du composant visuel de l'opérateur* `PlusGrand`

La figure 49 illustre le composant visuel de l'opérateur `PlusGrandÉgal`. Dans ce cas, seul l'exemple de droite illustre une situation où le résultat de l'opération est « faux » ; la valeur du terme `b` est supérieure à la valeur du terme `a`. Pour les deux autres exemples, la valeur est respectivement inférieure et égale.

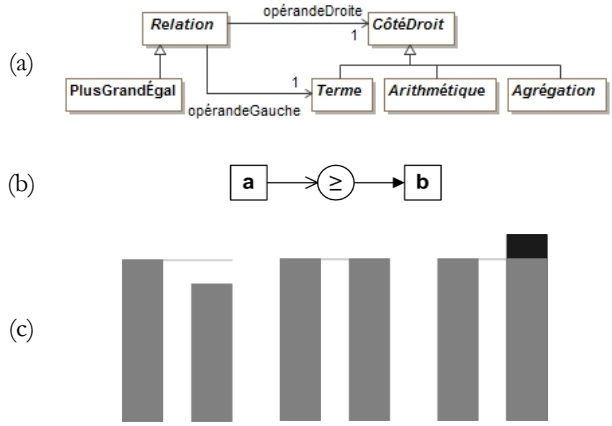


Figure 49. *(a) Extrait du méta-modèle, (b) représentation avec le langage WoDoMoLEID et (c) trois exemples du composant visuel de l'opérateur* `PlusGrandÉgal`

Les figures 49 et 50 illustrent respectivement le composant visuel de l'opérateur `PlusPetit` et le composant visuel de l'opérateur `PlusPetitÉgal`. Les

exemples suivent la même logique que celle des composants visuels des opérateurs `PlusGrand` et `PlusGrandÉgal`.

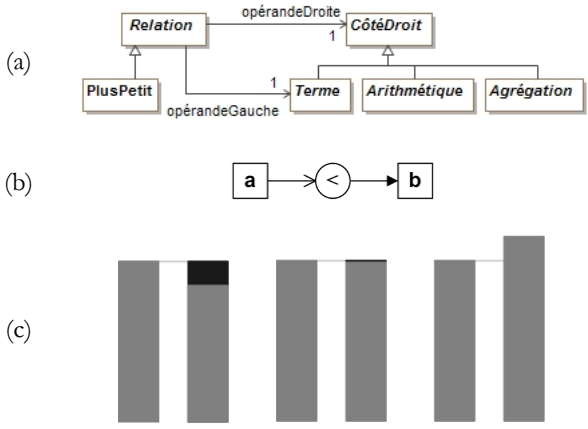


Figure 50. *(a) Extrait du méta-modèle, (b) représentation avec le langage WoDoMoLEID et (c) trois exemples du composant visuel de l'opérateur* `PlusPetit`

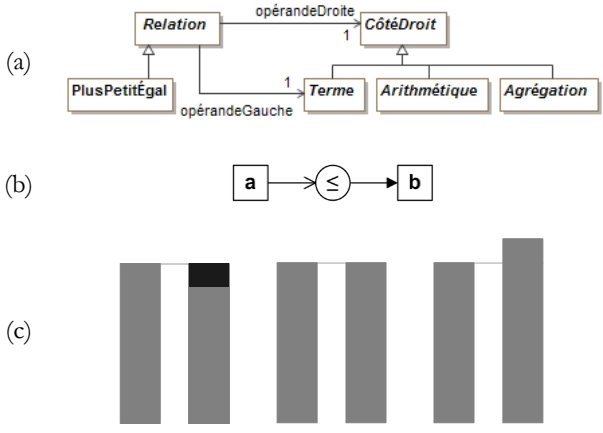
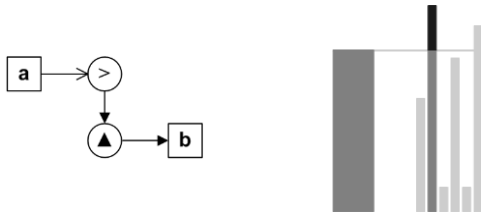


Figure 51. *(a) Extrait du méta-modèle, (b) représentation avec le langage WoDoMoLEID et (c) trois exemples du composant visuel de l'opérateur* `PlusPetitÉgal`

Les exemples précédents de composants visuels pour les opérateurs de relation utilisent tous le terme comme opérande de droite. Toutefois, tel que défini par le paquetage des opérateurs de relation du méta-modèle du langage WoDoMoLEID à la section 2.1.2.3, l'opérande de gauche peut être, en plus du terme, une opération d'agrégation ou une opération arithmétique.

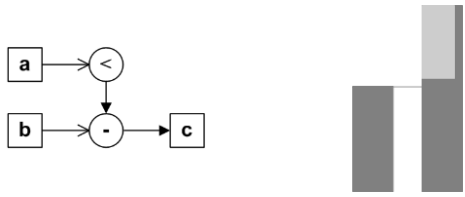
Les figures 51 et 52 illustrent deux exemples d'opération de relation avec, respectivement, une opération d'agrégation et une opération arithmétique. La figure 52 illustre un opérateur `PlusGrand` avec comme opérande de droite un opérateur `Max`. L'exemple illustre la situation où la valeur du terme `a` n'est pas supérieure au résultat de l'opérateur `Max`. La barre noire représente l'excédent. En ce qui concerne

la figure 53, elle illustre un opérateur `PlusPetit` avec comme opérande de droite un opérateur `Soustraction`. L'exemple illustre la situation où la valeur du terme `a` est inférieure au résultat de l'opérateur `Soustraction`.



(a) (b)

Figure 52. *(a) Représentation avec le langage WoDoMoLEID et (c) un exemple du composant visuel de l'opérateur* `PlusGrand` *avec un opérateur* `Max` *comme opérande de droite*



(a) (b)

Figure 53. *(a) Représentation avec le langage WoDoMoLEID et (c) un exemple du composant visuel de l'opérateur* `PlusPetit` *avec un opérateur* `Soustraction` *comme opérande de droite*