Intégration de métriques de qualité des diagrammes et des langages dans l'outil ModX

Xavier LE PALLEC
LIFL - Université Lille 1
Cité Scientifique, 59655 Villeneuve d'Ascq Cedex, France
xavier.le-pallec@univ-lille1.fr
Sophie DUPUY-CHESSA
LIG - Université Pierre Mendès France
B.P. 53, 38041 Grenoble Cedex 9, France
Sophie.Dupuy-Chessa@imag.fr

Abstract

Face au développement de nombreux langages spécifiques, leur qualité et celle des diagrammes qui en sont issus prennent une importance significative. Aussi nous nous posons la question de leur évaluation pour éventuellement les améliorer. Nous avons choisi une approche basée sur des métriques comme moyen d'évaluation autant des langages que des diagrammes. Nous souhaitons, en particulier, proposer des métriques portant sur la qualité des notations visuelles, aspect souvent négligé en ingénierie dirigée par les modèles. Cet article présente un premier pas vers de telles métriques en présentant les variables nécessaires à leur calcul ainsi que l'implémentation de l'une d'elles dans un outil de modélisation.
Mots-clés : qualité des diagrammes, qualité des langages, métriques, notations

1 Introduction

L'importance de l'utilisation des modèles est démontrée par le courant de recherche de l'ingénierie dirigée par les modèles, IDM, qui a connu ces dernières années un essor important. Avec l'IDM, de nombreux langages ont vu le jour pour prendre en compte des aspects spécifiques tels que les dispositifs d'interaction [3]. Les risques de ces nouveaux langages sont de produire des diagrammes inutiles ou incompréhensibles, ou de se baser sur des méta-modèles et des notations non exploitables. Nous nous intéressons en particulier, à un aspect de la qualité des langages de modélisation qui a été souvent été négligé jusqu'à présent : les syntaxes concrètes, en particulier celles graphiques. En effet, même si l'aspect visuel des diagrammes semble être de plus en plus perçu comme une préoccupation de premier ordre en ingénierie logicielle, peu de travaux [5, 4, 6, 2] et encore moins d'outils abordent l'évaluation de leur qualité. Notre travail se focalise donc sur la qualité des syntaxes concrètes. Pour cela, nous nous basons sur la physique des notations telle que définie par Moody [5]. De nombreux travaux [4, 6, 2] s'y réfèrent déjà et l'utilisent pour évaluer des langages. Cette physique des notations étant récente, il n'y a à l'heure actuelle aucune publication sur des calcul de métriques ou d'indicateurs qui y sont associés. Dans cette perspective, nous proposons ici une base logicielle pour ce type d'étude : établir la liste des fonctions élémentaires qu'impliquent les critères de la physique des notations et qu'un outil de méta-modélisation doit fournir afin de pouvoir définir de nouvelles métriques. Nous montrons ensuite comment ces fonctions ont donné lieu à une métrique implémentée au sein de l'outil de méta-modélisation, ModX.

Dans la section suivante, nous présentons les fonctions pour le calcul de métriques sur la qualité des langages. Dans la section 3, l'outil ModX est présenté avec l'automatisation partielle d'une métrique issue des critères de [5]. Enfin nous concluons par une synthèse et les perspectives de ce travail.

2 Fonctions élémentaires

A partir de [5], nous avons établi une première liste des propriétés concernant un langage qui constituent les données élémentaires pour le calcul des indicateurs de qualité.
Pour la syntaxe abstraite d'un langage, le calcul d'un indicateur nécessite un accès aux éléments abstraits. L'interface réflexive du MOF ou le paquetage ecore d'EMF sont suffisants pour cela. Concernant la syntaxe concrète, il convient d'avoir une interface équivalente à celle disponible pour la syntaxe abstraite. Toutefois, nous pouvons préciser les propriétés élémentaires nécessaires pour le calcul de métriques. En effet, Moody propose neuf critères pour évaluer l'efficacité cognitive d'une notation visuelle. Nous les reprenons ici en indiquant pour chacun d'eux la ou les fonctions d'accès lorsqu'elle(s) existe(nt) (synthèse dans le tableau 1). Pour la spécification des fonctions, l'étoile indique une liste de valeurs, et les accolades un objet dont les propriétés sont indiquées à l'intérieur séparées par une virgule. Les types de valeur sont volontairement omis car ils sont très dépendants du support/environnement utilisé.

Clarté sémiotique. C'est une relation bijective entre l'ensemble des éléments abstraits et l'ensemble des éléments concrets. Pour éviter la redondance, la surcharge, l'excès ou le déficit de symboles, il est conseillé d'avoir un et un seul élément concret pour chaque élément abstrait. Cet élément concret ne doit pas être lié à plusieurs éléments abstraits. Il est donc nécessaire ici d'avoir une fonction qui permette de connaître le ou les éléments concrets pour un élément abstrait donné (`élémentsConcrets ( stxAbs, stxCon, elmAbs ) = elmCon*`) et une fonction qui permette de connaître le ou les éléments abstraits pour un élément concret donné (`élémentsAbstraits ( elmCon ) = elmAbs*`). Bien sûr, il faut aussi une fonction qui permette d'avoir la liste des éléments concrets utilisés par une syntaxe concrète (`listeÉlémentsConcrets ( stxCon ) = elmCon*`).

Discrimination perceptive. Il s'agit du niveau de discrimination visuelle entre deux éléments concrets différents. Plus celle-ci est élevée, plus la perception des diagrammes sera rapide et plus le traitement cognitif qui suivra en sera donc facilité. La distance visuelle est la principale fonction de ce critère. Toutefois, il n'existe pas à l'heure actuelle de formule associée. Issue du domaine de la cartographie, cette fonction renvoie à plusieurs variables visuelles : la position (x,y), la taille, la valeur (clair vers foncé), le grain (ex : hachure, texture), la couleur, l'orientation, la forme. La distance visuelle entre deux éléments est en rapport avec le nombre de variables visuelles sur lesquelles les éléments diffèrent. Ces variables visuelles sont les propriétés dont l'accès est nécessaire pour le calcul de la discrimination perceptive. Ci-dessous nous présentons la liste des fonctions nécessaires à son calcul.

```
position ( elmCon ) = { x, y }      taille ( elmCon ) = { largeur, hauteur }
valeur ( elmCon ) = intensité       grain ( elmCon ) = matrice de motif
couleur ( elmCon ) = { r, g, b }    orientation ( elmCon) = angle
forme ( elmCon ) = description
```

Transparence sémantique. A la lecture d'un diagramme, l'intelligence perceptive du lecteur va attacher du sens à chaque élément visuel. Pour les expert(e)s d'une syntaxe concrète, le sens sera celui de la construction sémantique associée. Pour les autres, il est important que le sens que chacun d'eux associe à la représentation ne soit pas trop éloigné de l'élément abstrait associé. La transparence sémantique renvoie à cette distance sémantique. C'est un critère difficile à évaluer car il dépend de beaucoup de paramètres : profil du lecteur, contexte métier, pratiques concernant les symboles utilisés dans le domaine métier... L'évaluation de ce critère nécessite une étude comparative à faire à partir de corpus plutôt qu'à une formule à appliquer. Il n'y a donc pas ici d'accès particulier à des propriétés.

	Clarté sémiotique	Discrimination perceptive	Transparence sémantique	Gestion de la complexité	Intégration cognitive	Expressivité visuelle	Double codage	Économie graphique	Adaptation cognitive
élémentsConcrets	X							X	
listeÉlémentsConcrets	X								
élémentsAbstraits	X								
position		X				X		X	
taille		X				X		X	
valeur		X				X		X	
grain		X				X		X	
couleur		X				X		X	
orientation		X				X		X	
forme		X				X		X	
mécanismeComplexité				X					
représentationContexte					X				
annotationTextuelle							X		
supportDiagrammePrivilégié									X

Table 1: Critères et fonctions d'accès

Gestion de la complexité. La représentation d'un système complexe est une thématique de recherche transversale à de nombreuses disciplines. C'est aussi une problématique lorsqu'il s'agit de représenter des modèles complexes c'est-à-dire constitués de nombreux éléments et/ou nombreuses connexions. Gérer cette complexité est donc une condition sine qua none et implique des mécanismes dédiés dans les syntaxes concrètes. L'encapsulation graphique, la fragmentation en différents diagrammes sont des exemples de mécanismes possibles. Une fonction élémentaire intéressante à fournir pour ce critère est de pouvoir connaître le type de mécanisme associé à toute relation conteneur-contenu, type de relation la plus à même d'être support à la gestion de la complexité (`mécanismeComplexité (stxAbs, stxCon, relation contenant-contenu) = mécanismeGraphique*`).

Intégration cognitive. Lorsque le lecteur navigue dans les différents diagrammes via les mécanismes de gestion de complexité associés, il lui faut des éléments visuels lui permettant de se rappeler dans quel contexte se situe le diagramme qu'il/elle a sous les yeux. En d'autres termes, il faut pouvoir intégrer la partie du modèle étudié dans la représentation mentale du modèle global. Par exemple, pour une page web, il est classique d'afficher le chemin de la page actuelle au sein de la hiérarchique du site. Une fonction élémentaire pour évaluer ce critère est d'avoir l'élément visuel (pour l'intégration cognitive) correspondant à chaque mécanisme de gestion de complexité utilisé (`représentationContexte (stxAbs, stxCon, relation contenant-contenu) = elmCon*`).

Expressivité visuelle. Une syntaxe concrète est expressive visuellement si elle exploite un grand nombre de variables visuelles et utilise un grand nombre de valeurs pour chacune d'elles. Plus l'expressivité visuelle est grande, plus une syntaxe est efficace cognitivement. Ce critère utilise les mêmes propriétés que la discrimination perceptive.

Double codage. Même s'il n'est pas conseillé d'utiliser seulement du texte pour représenter un élément de modèle, l'annotation textuelle est par contre très conseillée pour renforcer une forme géométrique ou une icône. L'évaluation du double codage implique de connaître les annotations visuelles associées à l'élément abstrait (et non celles associées à l'une de ses propriétés) (`annotationTextuelle ( elmCon ) = annotation textuelle*`).

Économie graphique. Il convient de ne pas avoir un vocabulaire visuel trop important, c'est-à-dire d'utiliser un trop grand nombre de formes/liens différents. Pour une notation trop riche, l'activité mémorielle dédiée à l'association représentation-sens devient chez les lecteurs non-experts trop importante et gênera la lecture des diagrammes produits. Ce critère implique par exemple de partitionner les modèles en diagrammes de types différents. La clarté sémiotique est ici en défaut, car l'économie graphique peut impliquer, dans le cas de langages sémantiquement riches, un déficit de symboles dû au partitionnement. Il est nécessaire de disposer ici de la fonction permettant de connaître le nombre d'éléments concrets, résultat que l'on peut déduire d'une des fonctions nécessaires à la clarté sémiotique.

Adaptation cognitive. Les capacités de dessin sont en rapport avec le support utilisé pour le dessin des diagrammes. Par exemple, l'utilisation d'images complexes est peu viable lorsque les diagrammes se feront au stylo sur une feuille de papier. Parallèlement, le niveau d'expérience du lecteur concernant l'écriture de diagrammes de type ingénierie logicielle est aussi à prendre en compte. En effet, les différents mécanismes cités plus haut (comme la fragmentation en plusieurs diagrammes) demandent moins d'effort dans leur utilisation chez une personne expérimentée que chez un débutant. Une fonction pour connaître le choix (par le concepteur du langage) du support d'écriture peut être intéressante (`supportDiagrammePrivilégié ( stxAbs, stxCon ) = support`) mais elle implique que le concepteur fasse ce choix. Concernant le profil du lecteur, cela renvoie à des études comparatives comme pour la transparence sémantique et est hors de notre problématique de calcul de métrique.

L'évaluation de la qualité des syntaxes concrètes au travers de métriques implique de disposer de fonctions d'accès aux propriétés de celles-ci que nous venons de lister. Cette liste provient des critères définis dans la physique des notations. Nous cherchons à implémenter ces fonctions dans un outil de définition de langages afin d'automatiser ces métriques.

3 Automatisation partielle dans l'outil ModX

Nous avons implémenté les fonctions évoquées plus haut dans l'outil ModX [7]. ModX est un outil de modélisation et méta-modélisation créé à Lille en 2004 et basé sur la norme MOF (Meta-Object Facility de l'OMG). Initié dans le cadre du réseau d'excellence Kaleidoscope, cet éditeur a pour but de manipuler graphiquement tout type de modèle dans le domaine de l'Ingénierie Logicielle (e-Learning [1], IHM [8]). Il permet de créer des méta-modèles (syntaxe abstraite), d'y associer un ou des formalismes graphiques (syntaxe concrètes) et d'en éditer des instances, c'est-à-dire des modèles au travers de diagrammes. Le lecteur pourra se référer au site de ModX pour trouver une description détaillée du fonctionnement (http://www.lifl.fr/modx). Nous souhaitons offrir aux concepteurs de langages de modélisation un support pour mesurer la qualité de leurs syntaxes concrètes. ModX propose pour cela une interface de programmation (en Javascript) pour accéder aux syntaxes abstraites et concrètes définis dans ModX ainsi qu'à leurs diagrammes. Nous avons établi des correspondances entre les fonctions d'accès et les fonctions/propriétés présentes dans ModX pour montrer comment les fonctions d'accès pourraient être implémentées dans l'outil.

Ensuite nous avons aussi implémenté quelques calculs simples de métriques pour montrer la faisabilité de l'approche. Le script 1 permet de vérifier si la distance visuelle est suffisante

entre chaque paire d'éléments concrets (le script complet est présent dans la version 1.6 de ModX). Il est à noter que la distance visuelle dépend de son contexte d'utilisation, et que pour l'instant aucun travail n'a suffisamment effectué d'expérimentations pour définir une formule de calcul de cette distance dans le cadre des notations visuelles en Ingénierie Logicielle. Pour cette raison, l'exemple de script se veut très simplifié.

Pour chacune des paires, la fonction `computeForAll` (lignes 25-29) va calculer une distance visuelle. Si elle est supérieure à un 1 - dans l'idée qu'il y a plus d'une variable visuelle où les valeurs diffèrent - alors la distance est considérée comme suffisante. Sinon, le couple visuellement trop proche sera affiché. Pour les classes (lignes 2-15), la distance ne se calcule que si les deux éléments concrets utilisent des formes géométriques (lignes 4-5). Si au moins l'un des deux utilise une image, alors la distance vaut 2 (ligne 13) donc une distance suffisamment grande. Dans le cas contraire, on regarde si la forme géométrique choisie (ligne 6), la couleur de remplissage (ligne 7), la bordure (trame, ligne 8) ou la taille (si fixée, lignes 9-12) diffèrent. Pour les associations, la distance ne se calcule qu'entre éléments concrets ayant choisi des liens/traits graphiques (ligne 18). Dans ce cas, on regarde les différences entre les formes utilisées pour chacune des extrémités (lignes 19-20) et la trame du trait (ligne 21).

```
1  visualDistance = {
2    between2classes : function ( element1, element2) {
3      var difference = 0;
4      if (element1.formMode==element2.formMode &&
5        element1.formMode==concreteSyntax.SHAPE_MODE) {
6      difference+=(element1.shape!=element2.shape ? 1 : 0 );
7      difference+=(element1.backgroundColor!=element2.backgroundColor ? 1 : 0 );
8      difference+=(element1.borderType!=element2.borderType ? 1 : 0 );
9      if (element1.resizeMode==element2.resizeMode &&
10       element1.resizeMode==concreteSyntax.NO_RESIZE)
11       difference+=(element1.width!=element2.width ||
12         element1.height!=element2.height ? 1 : 0 );
13     } else difference = 2;
14     return difference;
15   },
16   between2associations : function (element1, element2) {
17     var difference=0;
18     if (element1.style==element2.style && element1.style==concreteSyntax.
           LINK_MODE) {
19       difference+=(element1.leftEndStyle!=element2.leftEndStyle ? 1 : 0 );
20       difference+=(element1.rightEndStyle!=element2.rightEndStyle ? 1 : 0 );
21       difference+=(element1.stroke!=element2.stroke ? 1 : 0 );
22     } else difference = 2;
23     return difference;
24   },
25   computeForAll : function (concreteSyntax) {
26     // utilise la fonction addMetric (titre, valeur, commentaires)
27     // pour afficher les couples d'elements concrets trop proches
28   }
29 }
```

Listing 1: Distance visuelle simplifiée pour une syntaxe concrète

4 Conclusion et Perspectives

Nous avons présenté une approche basée sur des métriques pour évaluer la qualité de la syntaxe visuelle des langages de modélisation et des diagrammes qui en sont issus. Les métriques sont intégrées dans un environnement de (méta-)modélisation pour faciliter le travail des concepteurs.

La faisabilité de notre approche ayant été montrée, il convient de vérifier sa pertinence en réalisant des expérimentations. Dans un premier temps, nous envisageons de valider auprès d'un large public, la métrique de calcul de la distance visuelle entre les éléments concrets d'une notation. L'objectif est de montrer que ce calcul automatisé aboutit à des résultats équivalents à ceux qu'auraient fourni intuitivement les utilisateurs. Ensuite, nous pourrons envisager des expérimentations auprès de concepteurs de langages et de diagrammes pour valider l'intérêt de disposer de telles métriques dans un environnement de (méta-)modélisation.

Un autre point important pour rendre ModX plus performant est d'enrichir l'ensemble des métriques proposées par défaut dans l'outil. Il nous semble particulièrement important d'approfondir celles relatives à la qualité des notations visuelles. En effet, elles nous semblent pertinentes pour des non-informaticiens qui seraient amenés à lire ou à modifier des diagrammes. ModX ayant pour ambition de s'adresser à ce public, il convient de disposer d'un ensemble de métriques capable de guider des utilisateurs non-experts en modélisation. Ces résultats pourraient ensuite être validés dans le cadre du projet ANR MOANO qui vise notamment à fournir des diagrammes appropriés à des botanistes.

5 Remerciements

Ce travail a été soutenu par l'ANR dans le cadre du programme CONTINT (Contenus numériques et interactions) au travers du projet MOANO (http://moano.liuppa.univ-pau.fr/).

References

[1] Pierre-André Caron, Mireille Blay-Fornarino, and Xavier Le Pallec. La contextualisation de modèles, une étape indispensable à un développement dirigé par les modèles ? *RSTI - Série L'Objet (RSTI-Objet)*, 13/4:55–71, 2008.

[2] Mario Cortes-Cornax, Sophie Dupuy-Chessa, Dominique Rieu, and Marlon Dumas. Evaluating choreographies in bpmn 2.0 using an extended quality framework. In *Proceedings o the 3rd International Workshop on the Business Process Model and Notation, BPMN 2011*, LNBIP. Springer-Verlag, 2011.

[3] Emmanuel Dubois, Philip Gray, and Laurence Nigay. Asur++: A design notation for mobile mixed system. In *In: Mobile HCI '02: Proceedings of the 4th International Symposium on Mobile Human-Computer Interaction. ISBN: 3-540-44189-1*, pages 123–139, 2002.

[4] Nicolas Genon, Patrick Heymans, and Daniel Amyot. Analysing the cognitive effectiveness ofthebpmn2.0visualnotation. In Brian A. Malloy, Steffen Staab, and Mark van den Brand, editors, *Software Language Eengineering, SLE'2010*, volume 6563 of *Lecture Notes in Computer Science*, pages 377–396. Springer, 2010.

[5] Daniel Moody. The physics of notations: Toward a scientific basis for constructing visual notations in software engineering. *IEEE Trans. Softw. Eng.*, 35(6):756–779, November 2009.

[6] Daniel Laurence Moody, Patrick Heymans, and Raimundas Matulevicius. Improving the effectiveness of visual representations in requirements engineering: An evaluation of i* visual syntax. *Requirements Engineering, IEEE International Conference on*, 0:171–180, 2009.

[7] Xavier Le Pallec, Emmanuel Renaux, and Cesar Olavo Moura. Modx - a graphical tool for mof metamodels. In *ECMDA-FA'2005 Tools Exhibition ECMDA-FA Open Source and Academic Tools*, 2005.

[8] José Rouillard, Jean-Claude Tarby, Xavier Le Pallec, and Raphael Marvie. From Meta-modeling to Automatic Generation of Multimodal Interfaces for Ambient Computing. *International Journal On Advances in Software*, 3(3 & 4):318–332, 2010.