Analyse des documents graphiques : une approche par reconstruction d'objets

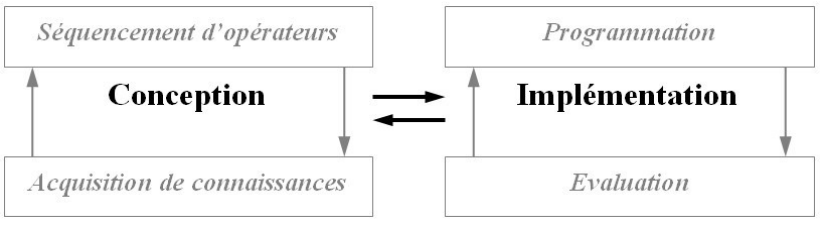
15 juin 2007

Annexe D : Interfaces Homme-Machine

Introduction

Comme nous l'avons illustré dans ce manuscrit la mise en oeuvre d'une application de reconstruction d'objets regroupe divers aspects : conception des opérateurs, définition de la stratégie de reconstruction, apprentissage, ... La réalisation d'une telle application est donc une tâche complexe, mais aussi dédiée (elle est spécifique à chaque application).

Une démarche permettant d'assister cette réalisation consiste à recourir à une méthodologie de développement. Il existe de nombreuses méthodologies de développement [André 01] (SADT, UML, ...), cependant celles-ci ne sont pas exploitables pour la réalisation d'applications de traitement d'images [Clouard 02]. En effet, la réalisation de ces applications repose d'avantage sur l'étude de la mise en oeuvre des opérateurs (ou méthodes) de traitement d'images, que sur celle de l'architecture logicielle des applications. Pour nos applications de reconstruction d'objets nous nous sommes plus spécialement basés sur une méthodologie cyclique en deux étapes. Nous illustrons cette dernière sur la figure suivante, nous la détaillons dans la suite de cette introduction.



Méthodologie cyclique de développement d'applications

Premièrement, une étape de conception est nécessaire avant toute implémentation [Cauchard 99]. Cette étape est fonction des types de document traité. Elle requiert la participation d'un expert en traitement d'images et d'un expert métier. Durant cette étape, l'expert en traitement d'images doit pouvoir adapter facilement des opérateurs aux documents traités. Cette adaptation concerne aussi bien le pilotage de ces opérateurs que l'acquisition des connaissances associées [Saidali 04]. Cette étape de conception se décompose alors selon un cycle en deux phases : séquencement des opérateurs / acquisition de connaissances. Durant la phase de séquencement, l'expert en traitement d'images recherche la meilleure combinaison d'opérateurs adaptée au problème traité. Afin de rechercher cette combinaison, cette phase se base sur l'utilisation d'Interface(s) Homme-Machine (IHM) pour le pilotage graphique des opérateurs [Clouard 99] [Saidali 04]. L'utilisation d'IHM permet, en effet, un contrôle aisé des opérateurs et une visualisation des résultats.

La seconde phase concerne l'acquisition des connaissances, en particulier graphiques [Yan 03]. En effet, afin de reconnaître les objets graphiques extraits, il est nécessaire d'acquérir préalablement les connaissances graphiques décrivant ces objets. Ces connaissances graphiques sont définies selon les représentations adoptées pour décrire ces objets (squelette, contour, région, ...). L'étape d' acquisition mettra donc en oeuvre les opérateurs d'extraction de primitives graphiques correspondant à ces représentations. Elle peut se réaliser à l'aide d'algorithmes d'apprentissage supervisé [Yan 03] ou non supervisé [Yan 04], ou par édition [Saidali 04]. Dans les deux cas les IHM permettent une acquisition aisée des connaissances graphiques [Yan 03] [Saidali 04].

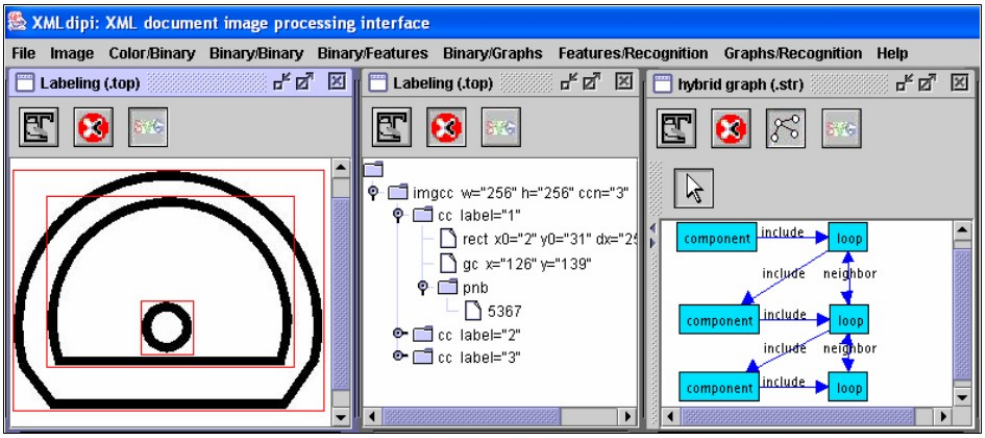
Une fois la conception terminée, l'expert en traitement d'image peut passer à l'étape d'implémentation de son application [Clouard 02]. Cette étape se décompose selon un cycle en deux phases : programmation / évaluation. La phase de programmation correspond à l'aspect génie logiciel de la réalisation de l'application[1]. La phase d'évaluation se réalise parallèlement à l'étape de programmation. En effet, celle-ci permet de tester la fiabilité des implémentations [Watkims 02]. Durant celle-ci, l'expert en traitement d'images utilise des données tests avec leur interprétation afin d'évaluer son application. Ces données et leur interprétation peuvent être obtenues par application de méthodes de bruitage sur des données modèles [Zhai 03], ou par saisie des vérités terrains à partir d'images scannées [Lopresti 02]. Cette dernière approche nécessite des IHM pour réaliser le processus d'acquisition.

Les IHM sont alors des éléments centraux dans le développement d'applications d'analyse des documents graphiques. Dans la suite de cette Annexe nous présentons nos IHM employées pour le développement de nos applications de reconstruction d'objets. Celles-ci sont employées à différentes étapes de notre méthodologie : conception graphique d'applications, acquisition des connaissances graphiques et saisie de vérités terrains. Nous présentons alors un dernier jeu d'interfaces pour la gestion de nos bases de connaissances.

[1] Nous reportons le lecteur à [Kernighan 99] sur ces aspects.

Conception graphique d'applications

Nous présentons dans cette section notre IHM pour la conception graphique d'applications : XMLdipi[2]. Celle-ci est basée sur l'utilisation de notre librairie PSI[3] exploitant nos différents opérateurs ainsi que notre formalisme objets pour leur interopérabilité. XMLdipi permet alors le pilotage de ces opérateurs, et la visualisation de leurs résultats. Elle est multi-fenêtres, chaque fenêtre correspond alors aux résultats d'un opérateur donné. Chacune des fenêtres est susceptible d'autoriser le déclenchement de un ou plusieurs opérateur(s) entraînant alors l'ouverture de nouvelle(s) fenêtre(s) résultat(s) dans XMLdipi. Ainsi un utilisateur peut revenir sur n'importe quelle étape de sa conception pour : adopter une nouvelle stratégie, confronter visuellement différents résultats, ...



XMLdipi

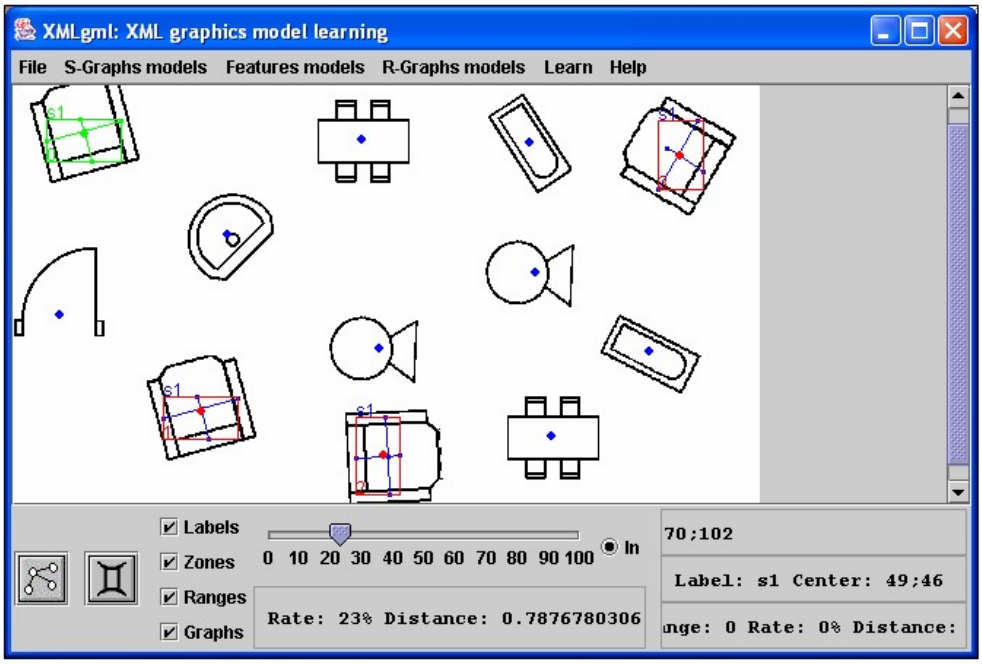
XMLdipi permet la manipulation de deux types de données : images (couleur, niveaux de gris, binaire) ou structurées (vectoriel, graphe, ...). Elle exploite pour cela deux panneaux de visualisation. Le premier utilise les fonctionnalités classiques des visualiseurs d'images (barre de déroulement, crop, pointeur de coordonnées, ...). La visualisation des images est basée sur un concept de tuilage permettant ainsi l'ouverture d'images de larges tailles. Le deuxième panneau permet la visualisation des données structurées exploitant un principe de multi-visualisation. Il utilise quatre types d'affichages de données structurées : données textuelles, données balisées (arbres XML), graphiques vectoriels (SVG)[4] et graphes (XGMML)[5]. Selon le formalisme employé pour les données de sortie d'un opérateur l'utilisateur peut naviguer entre ces différentes vues afin d'explorer les données extraites.

[2] XML document image processing interface
[3] Présentée en Annexe C de ce manuscrit.
[4] Basé sur la librairie Batik : http ://xml.apache.org/batik/
[5] Basé sur la librairie OpenJGraph : http ://openjgraph.sourceforge.net/

XMLdipi dispose ainsi de divers avantages par rapport aux IHM existantes pour la conception d'applications d'analyse des documents graphiques [Gomis 98] [Mejbri 02] [Clouard 02] [Saidali 04] [Rendek 04]. Elle permet, en effet, la visualisation de différentes catégories de données. Les IHM existantes se focalisent généralement sur les données images ou vectorielles. Elle permet également la conception d'applications de l'extraction des primitives jusqu'à l'étape de reconnaissance. Les IHM existantes se limitent généralement à l'étape d'extraction [Rendek 04], ou de reconnaissance [Witten 99].

Acquisition de connaissances graphiques

Nous présentons dans cette section notre IHM pour l'acquisition des connaissances graphiques : XMLgml[6]. XMLgml permet un apprentissage par l'exemple [Lieberman 01]. Cet apprentissage se réalise selon différents modèles de représentation (disponibles par défaut) de type statistiques (invariants Fourier-Mellin, moments de Zernike moments, caractéristiques géométriques, ...), et structurels (graphes de régions, graphes de squelette, ...). Le processus d'apprentissage avec XMLgml se déroule en plusieurs étapes.



XMLgml[7]

[6] XML graphic model learning

[7] L'objet graphique modèle et ceux similaires sont respectivement représentés en vert et rouge.

Tout d'abord l'utilisateur choisit un modèle de représentation parmi ceux disponibles dans XMLgml. Chaque modèle est fonction d'un opérateur implémenté au sein de XMLgml. Les opérateurs sont exploités de notre librairie PSI[8] et basés sur notre formalisme objets permettant leur interopérabilité. Ils sont implémentés dans XMLgml de façon similaire à notre système rsOPM[9]. En effet, XMLgml utilise pour cela un ensemble d'interface. Celles-ci sont des sur-couches logicielles servant à l'encapsulation des opérateurs. Les opérateurs peuvent être encapsulés individuellement ou composés afin de constituer des opérateurs plus complexes. Les interfaces sont externes à XMLgml et permettent le chargement dynamique des opérateurs lors de son ouverture. De cette façon XMLgml est aisément extensible et adaptable.

XMLgml permet ensuite l'extraction des primitives graphiques correspondantes au modèle de représentation sélectionné. Les primitives graphiques extraites sont exploitées par XMLgml sans retour sur les données d'entrée tout le long du processus d'apprentissage. À partir d'un objet graphique modèle (voir figure précédente), l'utilisateur effectue une recherche de similarité entre les objets graphiques. Cette recherche est basée sur l'utilisation de nos classifieurs[8] statistique et structurel. L'utilisateur peut ensuite décider de labelliser ces objets graphiques à la vue de leur distance de similarité avec l'objet modèle.

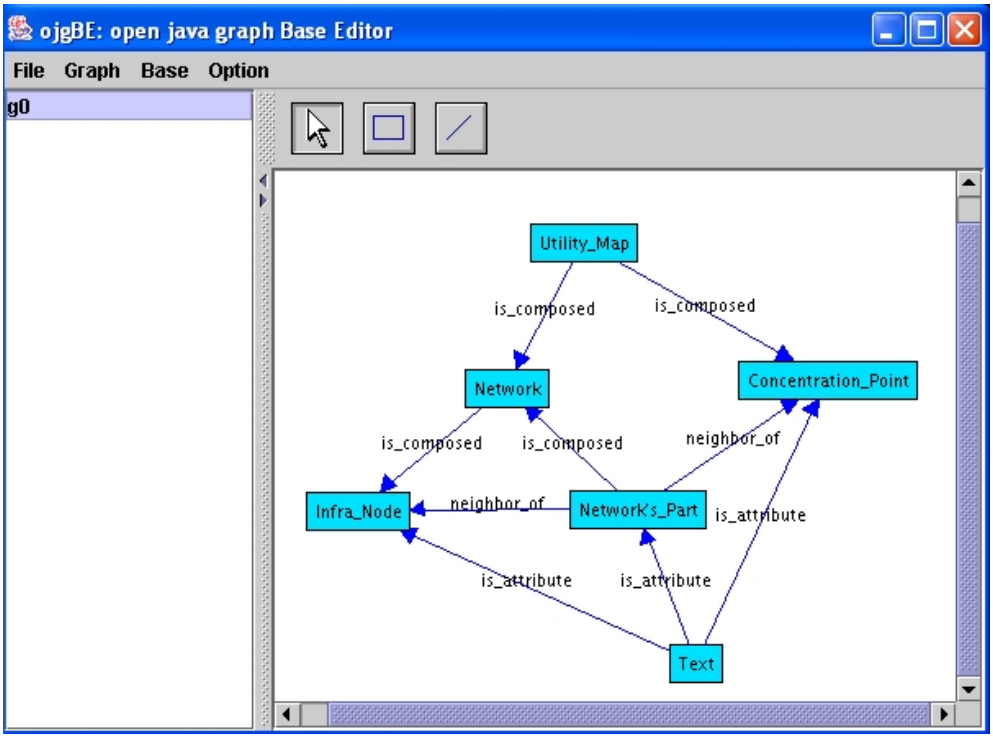
Durant le processus d'apprentissage, XMLgml permet de naviguer dans le schéma d'apprentissage en cours. Un schéma d'apprentissage correspond à un état de construction de la base d'apprentissage. La navigation s'effectue à l'aide d'une boîte de contrôle pour le filtrage des données du schéma (rang de similarité, rectangle englobant, label, ...) et d'une barre de distance pour le réglage de la similarité. À n'importe quel moment XMLgml permet d'analyser la carte de distances du schéma (distances min/max des différentes classes). À la vue de cette analyse, l'utilisateur peut décider d'effectuer un apprentissage arrière pour la remise en cause de ce schéma. De cette manière, l'utilisateur contrôle la robustesse de la base d'apprentissage. En effet, dans le cas d'objets graphiques a priori différents, mais proches selon le modèle de représentation utilisé, l'utilisateur peut décider de les regrouper dans une classe commune. L'intérêt est de réaliser un apprentissage en fonction du contexte représenté par les autres objets graphiques du document.

XMLgml présente divers avantages par rapport aux interfaces existantes dans la littérature [Saidali 04] [Yan 03] pour l'apprentissage de connaissances graphiques. En effet, XMLgml (basée sur nos opérateurs interopérables) permet la mise en oeuvre de différents modèles de représentation. Elle est de plus (aisément) extensible via l'utilisation des interfaces. Enfin, basée sur l'utilisation des schémas d'apprentissage XMLgml permet de tenir compte du contexte au cours du processus d'apprentissage.

[8] Présentée en Annexe C.
[9] Nous reportons le lecteur page **??**.

Nous avons développé une interface complémentaire à XMLgml pour l'acquisition des connaissances graphiques : ojgBE[10]. ojgBE permet à un utilisateur l'édition de graphes relationnels attribués. Plus précisément, les graphes sont attribués par des labels symboliques (ou textuels). ojgBE est basée sur l'utilisation de l'outil de modélisation et de visualisation de graphes OpenJGraph[11]. Ce dernier est basé sur l'utilisation du langage XGMML[12]. Via OpenJGraph, ojgBE permet d'éditer les arcs et/ou noeuds de graphes orientés et/ou non orientés. Les graphes édités sont regroupés au sein de bases. L'utilisateur peut par la suite gérer une base donnée afin d'y effectuer différentes actions sur les graphes : édition, copie, trie, suppression, ajout, ...

Il existe plusieurs interfaces d'édition de graphes[13]. Cependant aucune d'entre elles ne permet l'édition de base de graphes. Nous utilisons ojgBE en complément de XMLgml pour l'édition de graphes modèles sans recourir à des opérateurs d'extraction. Nous l'utilisons également pour l'acquisition de connaissances du domaine (ou métier) [Pasternak 95]. La figue suivante donne un exemple de graphe édité via ojgBE décrivant des relations du domaine relatives aux plans de réseau FT[14].

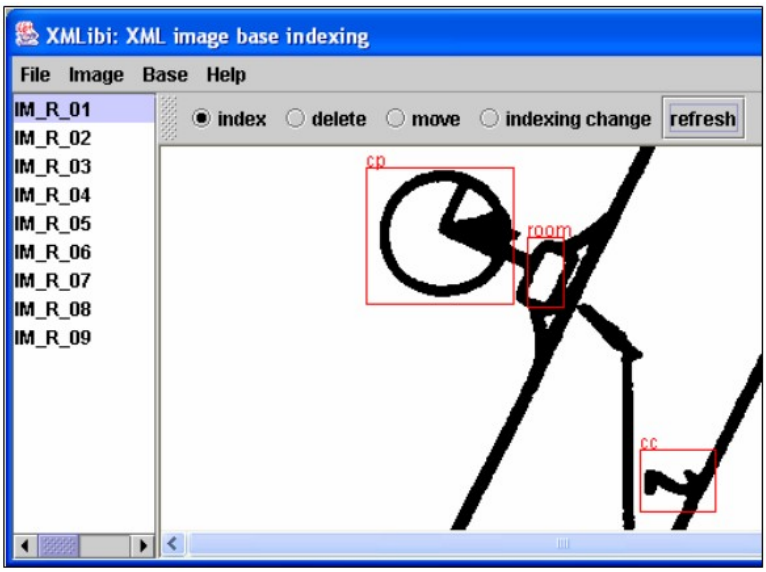


ojgBE

[10] open java graph Base Editor
[11] http ://openjgraph .sourceforge.net/
[12] eXtensible Graph Markup and Modelling Language
[13] http ://www.compendiumdev.co.uk/stareast2003/practical1_2.htm
[14] Nous reportons le lecteur page ??.

Acquisition de vérités terrains

Nous présentons dans cette section notre IHM pour l'acquisition de vérités terrains : XMLibi[15]. XMLibi permet l'annotation d'images par des utilisateurs. Les images annotées sont alors regroupées au sein de différentes bases afin de constituer des corpus de tests avec leurs vérités terrains correspondantes. Pour cela, l'utilisateur crée tout d'abord une base et y ajoute différentes images. Il peut par la suite naviguer dans cette base afin d'y visualiser les différentes images et les annoter. L'annotation d'une image se fait par édition de zones rectangulaires[16] puis par labélisation de ces zones. L'utilisateur peut à tout moment re-éditer ces zones, les déplacer ou les supprimer.



XMLibi

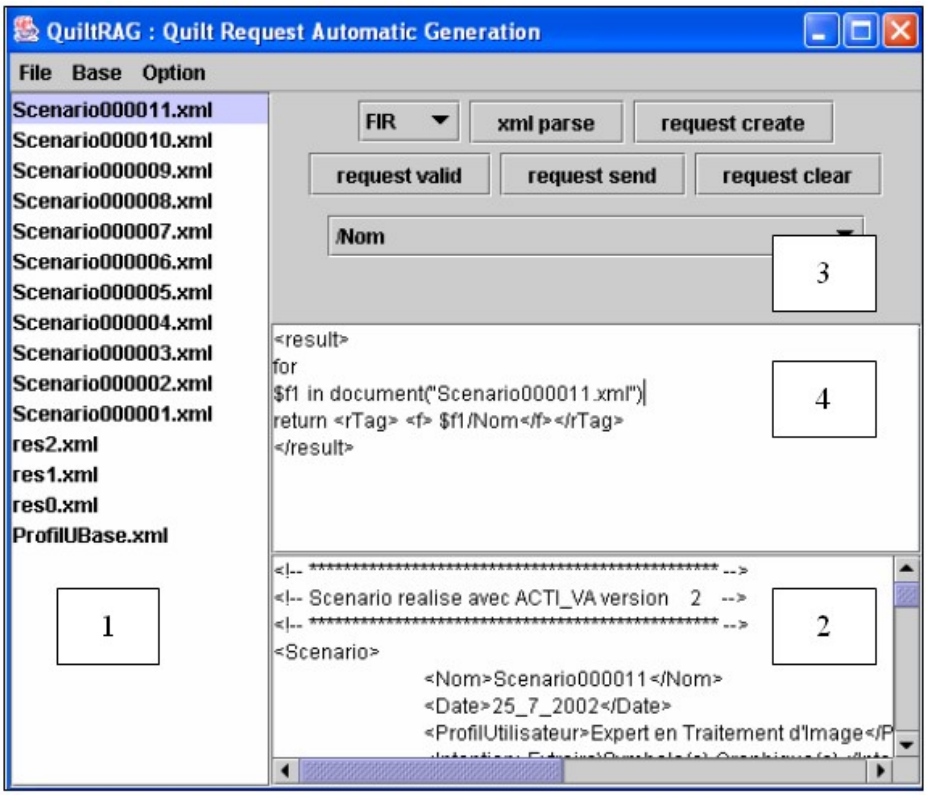
Gestion des bases de connaissances

Les connaissances au sein des systèmes d'analyse des documents graphiques sont peu structurées. En effet, elles sont fragmentées en différentes unités selon les différentes parties du système qui les utilisent. Cependant, il existe différents liens implicites entre ces connaissances (images et primitives extraites, primitives extraites et opérateurs associés, ...). Dans cette section nous présentons une approche pour la gestion des bases de connaissances permettant la restauration de ces liens. Celle-ci est basée sur l'utilisation d'IHM pour la navigation et la structuration des bases de connaissances.

[15] XML image base indexing
[16] bounding box

Notre approche est plus particulièrement basée sur l'utilisation de deux IHM : QuiltRAG et 2iRDF. QuiltRAG est utilisée elle pour la gestion des données semi-structurées. 2iRDF est utilisée, elle, pour la structuration des connaissances et la navigation dans les bases au travers de graphes de ressources. De cette façon, QuiltRAG et 2iRDF sont respectivement utilisées pour la gestion basée contenu et basée structure. Via ces deux IHM l'utilisateur peut naviguer, structurer, et explorer les bases de connaissances. Il peut ainsi gérer les bases de selon son point de vue et y restaurer les liens implicites existants. Nous présentons chacune de ces IHM dans la suite de cette section.

QuiltRAG[17] est une IHM pour la génération automatique de requêtes Quilt. Quilt [Chamberlin 00] est un langage candidat pour la norme XML-QL[18]. Ce langage décrit des requêtes de type FLoWeR[19]. Il utilise des expressions XPath dans le but d'adresser les noeuds des documents XML. Ansi, Quilt se veut être un langage particulièrement adapté pour l'exploration et la transformation des données XML [William 02].

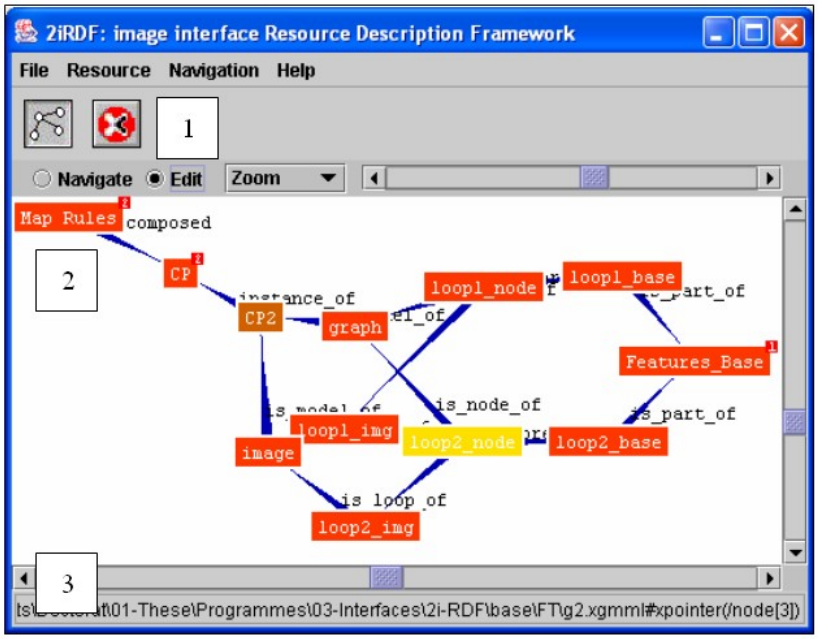


QuiltRAG

[17] Quilt Request Automatic Generation
[18] XML-Query Language
[19] For Let Where Return

QuiltRAG permet l'exploitation des requêtes Quilt via le moteur de requête Kweelt[20]. Dans une première étape, l'utilisateur ouvre une base de fichiers XML à partir d'un répertoire racine. Il peut ensuite (voir figure précédente) naviguer dans cette base à partir d'une liste de sélection [1], et afficher le contenu du document XML sélectionné dans un panneau de visualisation [2]. QuiltRAG permet alors de parser ce fichier XML afin d'en extraire les noms de tags et les chemins XPath. L'utilisateur exploite alors ces données dans le but de créer des requêtes de façon automatique [3]. Les requêtes créées sont alors affichées dans un panneau d'édition de requête [4]. L'utilisateur peut éditer les requêtes, les sauver, en charger, et les exécuter via le moteur Kweelt[20]. Les résultats XML d'exécution des requêtes sont alors ajoutés à la base de fichiers XML en cours. L'utilisateur peut ensuite explorer les fichiers XML résultats à travers la liste de sélection [1]. Ainsi, les résultats des requêtes enrichissent la base de fichiers XML. L'utilisateur peut via QuiltRAG gérer la base de fichiers XML par différentes actions : déplacement, suppression, copie, trie, ...

2iRDF[21] permet à un utilisateur de naviguer et de structurer des bases de connaissances. Via 2iRDF, l'utilisateur peut construire un graphe de ressources. Ces ressources correspondent à des fichiers de documents (image, XML, texte, ...). Le graphe de ressources est représenté au format RDF[22].



2iRDF

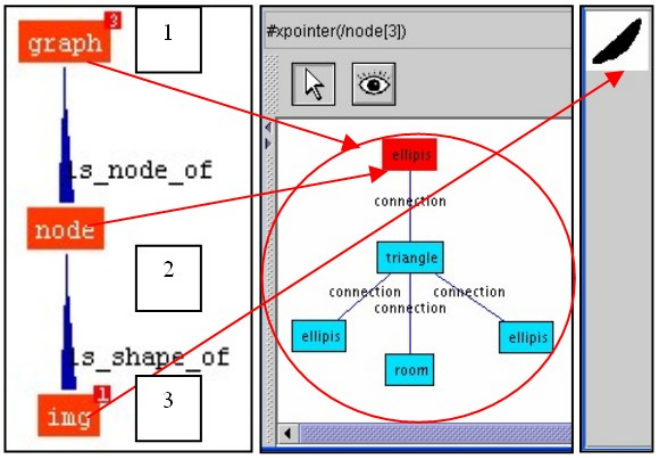
[20] http ://kweelt.sourceforge.net/
[21] interface for image Resource Description Framework
[22] Resource Description Framework

Dans une première étape, l'utilisateur peut ouvrir une base de fichiers à partir d'un répertoire racine [1] (figure précédente). Les fichiers de cette base sont chargés dans le graphe de ressources [2], chaque fichier correspond alors à un noeud. Le panneau de visualisation du graphe de ressources est basé sur l'application TouchGraph[23]. Celle-ci permet une visualisation dynamique des graphes : l'utilisateur peut alors naviguer dans un graphe de large dimension (plusieurs centaines de noeuds) selon des critères de localité sur le noeud sélectionné. Une barre d'état [3] donne le type du fichier ressource du noeud sélectionné, ainsi que son chemin. L'utilisateur peut visualiser ce fichier ressource en cliquant sur le bouton de visualisation de ressources (bouton [**X**] sur la barre [1]). Suite à cette action, le panneau de visualisation du graphe de ressources [2] est substitué par un panneau de visualisation de ressources. Ce panneau de visualisation de ressources permet différents types d'affichage selon le type du fichier ressource : image, arbre XML, graphique vectoriel (SVG)[24] et graphe (XGMML)[25].

Du panneau de ressources, l'utilisateur peut extraire des expressions XPath des données XML, XGMML et SVG. Ces expressions sont alors ajoutées comme nouvelles ressources dans le graphe de ressources par création de nouveaux noeuds. L'utilisateur peut par la suite structurer ces nouveaux noeuds dans le graphe de ressources via le panneau de visualisation de graphe. Il peut par exemple ajouter de nouveaux arcs labellisés dans le graphe de ressources entre les différents noeuds. Il peut également créer des noeuds sans ressources associées, enrichissant ainsi la description sémantique du graphe de ressources. Ainsi, l'utilisateur peut structurer des connaissances correspondant à des formalismes hétérogènes, et exploitées a priori à des niveaux différents dans une application. La figure suivante donne un exemple de structuration d'un graphe avec des primitives graphiques images correspondant aux occlusions d'un symbole : le noeud $\{graph\}$ [1] donne un lien sur un fichier XGMML, le noeud $\{node\}$ [2] donne un lien via une expression XPath sur un noeud du graphe XGMML, et le noeud $\{img\}$ [3] donne un lien sur un fichier image d'une occlusion. Ces trois noeuds ont été structuré dans le graphe de ressources avec deux arcs $\{is\_node\_of\}$ et $\{is\_shape\_of\}$.

Dans le but d'aider un utilisateur à naviguer dans les graphes de ressources de large dimension, 2iRDF permet d'exécuter des requêtes RDF-QL[26]. L'exécution de ces requêtes est basée sur l'utilisation du moteur de requête Jena[27]. À partir de 2iRDF, l'utilisateur peut afficher un panneau d'édition de requête RDF-QL. Ce panneau permet d'éditer, de sauvegarder, et d'exécuter les requêtes. À la suite de l'exécution d'une requête, le sous-graphe de ressources correspondant au résultat de la requête est affiché dans le panneau de visualisation de graphe. Le reste du graphe de ressources a été ainsi rendu invisible. Il peut être restaurer à n'importe quel moment par changement des critères de localité de l'application TouchGraph.

[23] http ://www.touchgraph.com/
[24] Basé sur la librairie Batik : http ://xml.apache.org/batik/
[25] Basé sur la librairie OpenJGraph : http ://openjgraph.sourceforge.net/
[26] RDF Query Language
[27] http ://jena.sourceforge.net/



Panneau de visualisation de ressources

Bibliographie

[André 01] P. André & A. Vailly. Conception des systèmes d'information. Editions Ellipses, ISBN : 272980479X, 2001.

[Cauchard 99] V. Ficet Cauchard. *Réalisation d'un Système d'Aide à la Conception d'Application de Traitement d'Images : une Approche Fondée sur le Raisonnement à Partir de Cas.* Thèse de Doctorat, Université de Caen, France, 1999.

[Chamberlin 00] D. Chamberlin, J. Robie & D. Florescu. *Quilt : An XML Query Language for Heterogeneous Data Sources.* In Workshop on the Web and Databases (WebDB), volume 1997 of *Lecture Notes in Computer Science (LNCS)*, pages 1–25, 2000.

[Clouard 99] R. Clouard, A. Elmoataz, C. Porquet & M. Revenu. *Borg : A Knowledge Based System for Automatic Generation of Image Processing Programs.* Pattern Analysis and Machine Intelligence (PAMI), vol. 21, no. 2, pages 128–144, 1999.

[Clouard 02] R. Clouard, A. Elmoataz & M. Revenu. *Une Méthodologie de Développement d'Applications de Traitement d'Images.* In Congrès Francophone de Reconnaissance de Formes et Intelligence Artificielle (RFIA), pages 1033–1042, 2002.

[Gomis 98] J.M. Gomis, P. Company & M.A. Gil. *Vectorization in Recovering Engineering Drawings.* In Seminario Diseño y Fabricabilidad de los Productos Industriales, pages 253–263, 1998.

[Kernighan 99] B.W. Kernighan & P. Pike. The practice of programming. Addison Wesley Publishing, ISBN : 0-201-61586-X, 1999.

[Lieberman 01] H. Lieberman. Your wish is my command : Programming by example. Morgan Kaufmann Editor, ISBN : 1558606882, 2001.

[Lopresti 02] D. Lopresti & G. Nagy. *Issues in Ground-Truthing Graphic Documents.* In Workshop on Graphics Recognition (GREC), volume 2390 of *Lecture Notes in Computer Science (LNCS)*, pages 46–66, 2002.

[Mejbri 02] E.F. El Mejbri, H. Grabowski, H. Kunze, R.S. Lossack & A. Michelis. *3D Reconstruction of Paper Based Assembly Drawings : State of the Art and Approach.* In Workshop on Graphics Recognition (GREC), volume 2390 of *Lecture Notes in Computer Science (LNCS)*, pages 1–12, 2002.

[Pasternak 95] B. Pasternak & B. Neumann. *The Role of Taxonomy in Drawing Interpretation.* In International Conference on Document Analysis and Recognition (ICDAR), volume 2, pages 799–802, 1995.

[Rendek 04] J. Rendek, G. Masini, P. Dosch & K. Tombre. *The Search for Genericity in Graphics Recognition Applications : Design Issues of the Qgar Software System.* In Lecture Notes in Computer Science (LNCS), volume 3163, pages 366–377, 2004.

[Saidali 04] Y. Saidali, S. Adam, J.M. Ogier, E. Trupin & J. Labiche. *Knowledge Representation and Acquisition for Engineering Document Analysis*. In Wokshop in Graphics Recognition (GREC), volume 3088 of *Lecture Notes in Computer Science (LNCS)*, pages 25–36, 2004.

[Watkims 02] J. Watkims. Test logiciel en pratique. Editions Vuibert Informatique, ISBN : 2711748065, 2002.

[William 02] K. William & al. Xml et les bases de données. Editions Eyrolles, ISBN : 2212092822, 2002.

[Witten 99] I.H. Witten & E. Frank. Data mining : Practical machine learning tools and techniques with java implementations. Morgan Kaufmann Editor, ISBN : 1-55860-552-5, 1999.

[Yan 03] L. Yan & L. Wenyin. *Engineering Drawings Recognition Using a Case-based Approach*. In International Conference on Document Analysis and Recognition (ICDAR), pages 190– 194, 2003.

[Yan 04] X. Yan, P.S. Yu & J. Hany. *Graph Indexing : A Frequent StructureBased Approach*. In Conference of Management of Data, pages 335–346, 2004.

[Zhai 03] J. Zhai, L. Wenyin, D. Dori & Q. Li. *A Line Drawings Degradation Model for Performance Characterization*. In International Conference on Document Analysis And Recognition (ICDAR), pages 1020–1024, 2003.

Table des figures

Table des matières