Une interface de programmation visuelle pour la composition de services de visualisation d'information

Romain Vuillemot, Béatrice Rumpler

Université de Lyon, CNRS
INSA-Lyon, LIRIS, UMR5205
F-69621, France
romain.vuillemot@insa-lyon.fr

RESUME

Dans cet article, nous nous intéressons à la création et au partage de visualisations d'information. Notre approche est de considérer la visualisation d'information comme le résultat d'un flot de traitement de données, constitué d'un assemblage de services web qui ont vérifié des règles syntaxiques et sémantiques. Afin de faciliter la composition de ces services, et donc de créer des visualisations d'information, nous introduisons *mashviz*, une interface de programmation visuelle destinée aux concepteurs de visualisations, ainsi qu'aux utilisateurs pour le partage et l'annotation de ces visualisations. Nous discutons les premières visualisations créées et leur partage, puis donnons les prochaines étapes de nos travaux, notamment l'évaluation par l'usage.

MOTS CLES : Visualisation d'Information, Programmation visuelle, Services Web, Profil Visuel Utilisateur.

ABSTRACT

In this article, we are interested in information visualisations creation and sharing. Our approach is to consider information visualisation as a dataflow, issued from web services compositions which held both syntaxic and semantic rules. To ease service composition, we introduce *mashviz*, a visual programming interface aimed to both designers and users, to share and annote visualisations. We discuss our early created visualisations, and give clues about our next steps, such as evaluation by usage.

CATEGORIES AND SUBJECT DESCRIPTORS: H.5.3 Information Interfaces and Presentation : Group and Organization Interfaces – *Web-based interaction*

KEYWORDS: Information Visualisation, Visual programming, Web Services, User Visual Profile.

IHM 2009, 13-16 Octobre 2009, Grenoble, France

INTRODUCTION

La visualisation d'information est un processus impliquant le traitement des données, depuis leur source, jusqu'à leur visualisation. Ainsi des compétences variées et pointues (ETL, IHM, Graphe, etc.) sont requises, issues de plusieurs disciplines, aussi bien en termes de Recherche, que d'applications. Le *pipeline* de visualisation d'information est donc relativement long et complexe. Celui-ci a déjà fait l'objet de décomposition, comme le *Data State Reference Model* introduit en 1999 par Ed Chi [1]. Ce modèle offre un cadre analytique, qui permet une meilleure compréhension des techniques de visualisation, sans pour autant permettre de facilement les recomposer. Un autre aspect essentiel de la visualisation est l'association d'attributs de données aux codes visuels. Là encore, de nombreux travaux ont été réalisés, notamment en 1986 par Mackinlay [9]. Ce dernier décrit le processus automatique d'association, selon le type de données (numérique, ordonée, etc.) et l'attribut visuel (couleur, position, etc.), qui, comme une phrase à composer, doit répondre à des règles syntaxiques et sémantiques. Ainsi, ces deux exemples montrent déjà que la combinatoire de création de visualisations *statiques* explose très rapidement, et dès qu'elles deviennent *interactives*, alors de nombreuses nouvelles visualisations cohérentes et synchronisées sont encore à générer.

Motivation Notre motivation est apparue dans un premier temps lors de la mise au point de différentes applications de visualisation d'information[1]. Nous nous sommes vite rendus compte, à la vue du vaste espace de conception, qu'il était nécessaire d'impliquer les utilisateurs afin de leur permettre d'intégrer leurs préférences dans toute la chaine de traitement de données. Une démarche scientifique était également nécessaire pour permettre une évaluation des résultats : facilement formuler et valider des hypothèses, permettre des expériences reproductibles, etc. Dans un deuxième temps, lors de démonstrations et discussions en internes au LIRIS[2], de nombreux chercheurs ont souhaité vouloir les tester avec leurs "mouli-

[1] `http://vizod.liris.cnrs.fr/projects/`
[2] `http://liris.cnrs.fr/`

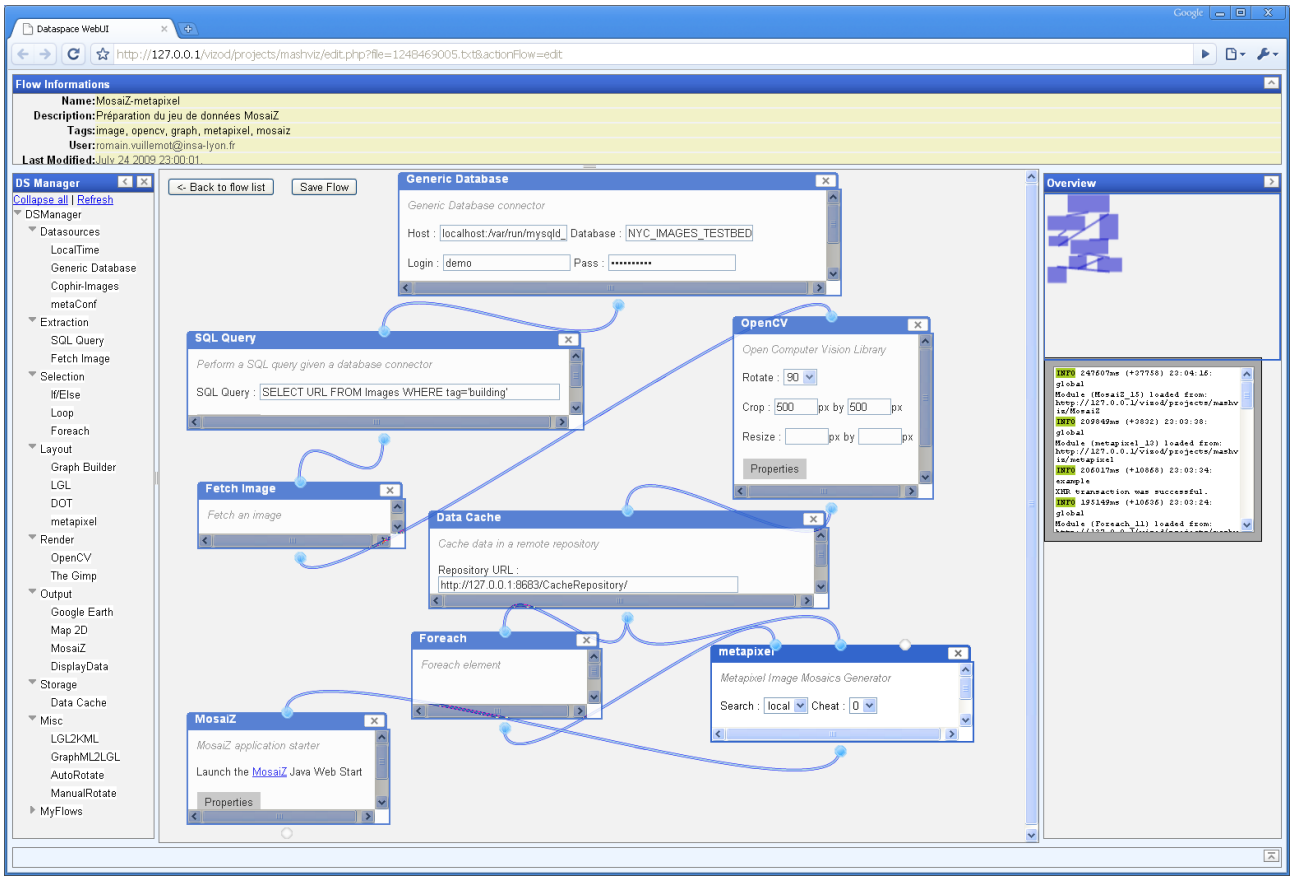

FIG. ***1*** **:** Capture d'écran de l'interface de création de flots de *mashviz*. Au centre, l'espace de composition de flots (séquence de boîtes reliées entre elles par des connexions). Le bandeau en haut regroupe les informations sur le flot en cours de création. La partie gauche regroupe la *base de services* organisée en catégories. Sur la partie droite est disponible une vue globale du flot ainsi que les logs d'informations (erreurs, ..).

nettes" (indexation, extraction de contours, etc.), même si le cœur de métiers de chacun est la base de données ou l'image : ils souhaitaient intégrer leurs contributions dans un environnement existant complet, jusqu'à la visualisation interactive. Il était donc nécessaire d'offrir un cadre limitant l'interdépendance, car les individus sont distants et peu disponibles ; ils souhaitent aussi garder leur langage de développement.

Approche Notre première approche technique [13] a été de séparer les étapes de traitements des données sous forme de services web, qui seront recomposés pour former un flot de visualisation d'information. Les services (qui ne seront pas abordés dans le cadre de cet article) reprennent le paradigme de conception objet en encapsulant le code et la complexité, pour ne laisser transparaitre qu'une interface. Cette approche est facilitée par les récents travaux de Fielding [4] qui ont permis de formaliser un style d'architecture d'accès aux services web, dit REST, et qui les rend disponibles sous forme de ressources accessibles via une URL (et non sensibles aux proxies). Pour encapsuler les services nous nous basons sur une API de description de services [10] qui permettra la création bas niveau de flots de données. Le but de cet article est désormais de décrire l'interface qui permettra à des utilisateurs de réaliser et partager ces compositions de services. L'interface est de type programmation visuelle de données qui est très bien adapté aux flots [8], et a déjà par exemple permis de faciliter la résolution de problèmes complexes comme avec V4Miner [2] pour la fouille de donnée, ou la boîte à outils ICON (Input Configurator [3]) de facilement relier les dispositifs d'interaction et de les adapter.

INTERFACE MASHVIZ

Nous décrivons *masvhiz*, une interface de programmation visuelle pour la composition de services de visualisation d'information. Tout d'abord nous introduisons brièvement quelques règles de composition de services de visualisation d'information.

Règles de composition de services Les services n'étant caractérisés que par leurs entrées et sorties, il est nécessaire d'y appliquer des règles de composition. Pour être composés, les services doivent valider deux niveaux de règles, dont nous donnons quelques exemples :

1. Des règles *syntaxiques* afin de valider la composition entre les services. Ces règles sont implémentées au niveau technique d'encapsulation des flots [10], afin de

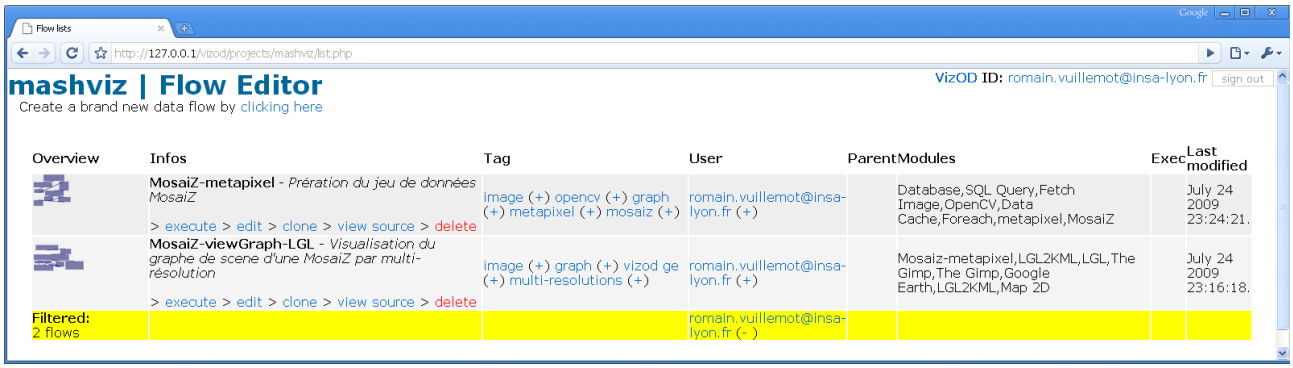

FIG. *2* **:** Capture d'écran de l'interface d'annuaire de flots de *mashviz*. Les flots déjà créés sont visibles sous forme de liste qu'il est possible de filtrer par tags ou par utilisateurs. Les utilisateurs peuvent choisir un flot existant pour l'éditer, le cloner ou l'utiliser comme une source de données (et l'inclure dans un nouveau flot). Les deux flots présentés ci-dessus sont utilisés dans le cadre de cet article pour respectivement pré-traiter un jeu de données d'images et visualiser un graphe.

garantir la continuité du flot. Par exemple la *règle de transitivité* permet de remplacer une séquence de flots en un nouveau flot qui apparait dans la liste de flots. Egalement une *règle d'équivalence* permet de remplacer un flot par un autre, basée sur une similitude d'entrées et de sorties.

2. Des règles *sémantiques* afin de vérifier la validité conceptuelle de la composition entre les services. Elles permettent de vérifier la validité à haut niveau de la composition de services, en phase avec les recommandations dans le domaine de la visualisation d'information et de l'interaction homme/machine. Par exemple une *règle historique* doit assurer qu'un flot permet une transition vers le flot précédent. Egalement une *règle de consistance* doit assurer que les flots au fil du temps gardent la même métaphore visuelle.

Présentation générale de l'interface Notre interface *mashviz* intègre les règles précédemment énoncées et se présente sous forme d'une interface de programmation visuelle (figure 1). Côté ergonomie, *mashviz* est fondée sur le *look and feel* de Yahoo ! Pipes [5] pour l'apparence générale, l'encapsulation des services en boîtes, et la connexion élastique par *tuyaux* en courbes de Bézier. L'interface possède une partie de création de flots (figure 1), l'autre d'exploration de flots existant à partir d'un annuaire de flots (figure 2).

Interface de création de flots (figure 1) L'utilisateur possède un plan de travail dans lequel des services peuvent être glissés/déposés depuis la *base de service*, qui a été divisée en catégories d'étapes de traitement des données [13]. Chaque service déposé peut être personnalisé dans la boîte qui le représente. Cette boîte est publiée par le service lui même et est affichée sous forme de sous-page web. Ensuite, l'utilisateur peut connecter/déconnecter les entrées et sorties des services afin de créer un *flot de services*. Il n'est pas possible pour l'utilisateur de relier des boîtes incompatibles. La compatibilité étant basée sur une similarité de typage similaire à `MIME` (Multipurpose Internet Mail Extensions). Enfin, le flot ainsi créé peut être sauvegardé et restauré par la suite au moyen de l'interface d'annuaire de flots.

Interface d'annuaire de flots (figure 2) L'interface d'annuaire de flots permet de visualiser les flots déjà créés et d'obtenir des informations sur ces flots, telles que : la vue globale du flot (image miniature qui donne une information quantitative), le nom, la description, les tags (qui incluent les services qui les composent), l'utilisateur et les dernières modifications. Il est possible d'éditer un flot ou de le cloner (le lien de parenté sera sauvegardé) en cliquant dessus. Enfin, des fonctionnalités de filtrage et de recherche de flots sont fournies, ce qui permet par exemple de trouver les flots qui utilisent un certain service ou qui ont été créés par un utilisateur en particulier.

Couplage aux applications interactives Afin d'intégrer les flots à une application interactive, il est nécessaire de coupler de manière flexible ces flots au contrôleur de l'application. Pour cela, les flots sont à tout moment disponibles sous forme de ressource unique type `URL`, à laquelle peuvent être passés des paramètres issus de widgets de sélection de liste, choix multiples, etc. Tout environnement ou widget applicatif peut ainsi devenir contrôleur, par exemple un slider pour les variables numériques, ou l'altitude sur un globe terrestre peut permettre la multi-résolution [11]. A noter qu'il est également possible d'obtenir plusieurs vues multiples coordonnées sur un même flot de données, en connectant différents sorties à un même flot (chaque sortie aura sa propre URL).

Détails d'implémentation de *mashviz*. Nous avons choisi une interface type web pour élargir au maximum l'utilisation de l'interface. Nous avons utilisé la bibliothèque Yahoo ! User Interface Library (YUI) [7] et WireIT [6]. Le serveur web est Apache couplé à une base de donnée pour effectuer des requêtes sur le choix des flots.

PREMIERS RESULTATS ET PERSPECTIVES

Application à l'exploration d'images et de graphes. Parmi nos projets de visualisation d'information, nous avons souhaité étudier la métaphore des mosaïques d'images pour l'exploration visuelle de grandes collections non-structurées d'images [12]. Nous avons vite constaté que l'espace de conception était grand et qu'il était nécessaire de laisser l'utilisateur varier les associations possibles, et surtout au niveau des étapes de traitement de données qui influent sur la pertinence des mosaïques. Notre interface *mashviz* nous a permis de séparer les étapes de traitement qui étaient au par avant programmées en dur ou sous formes de scripts. Désormais tout utilisateur peut comprendre facilement l'anatomie de l'application (illustrée figure 1) et la modifier.

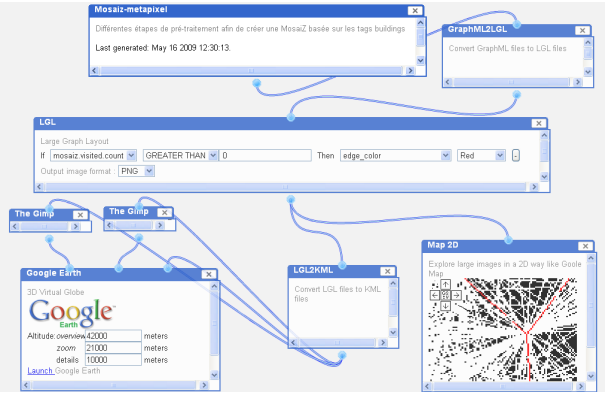

FIG. ***3*** **:** Comme premier résultat de l'utilisation de l'interface, un utilisateur peut via quelques clics seulement ré-utiliser la préparation du jeu de données d'une application, vers une autre application (ici le graphe de scène des MosaiZ est désormais exploré par multi-résolution).

Résultats La figure 3 montre que l'utilisateur a déconnecté la sortie du graphe de scène interne aux MosaiZ, pour effectuer une navigation multi-résolution [11]. L'utilisateur a ainsi au moyen de quelques clics réutilisé le flot `Mosaiz -metapixel` existant en y accédant via la catégorie `MyFLows` de sa *base de service* de l'interface de création (visible en bas de l'arborescence du panneau gauche). Ainsi, pour un même flot de données (la sortie MosaiZ a cependant été supprimée), une autre de nos applications a pu être utilisée sans la modifier (uniquement le contenu a été changé, à savoir l'`URL` à laquelle se connecter). L'utilisateur a aussi facilement changé l'association attribut données/code visuel, en coloriant les sommets visités du graphe en rouge (visités via l'utilisation de l'application MosaiZ) ce qui aurait normalement nécessité des compétences de programmation avancées et du temps.

Discussions et perspectives. Disposant désormais d'un cadre technique complet (de l'extraction des données à la visualisation, de la définition de besoins à l'évaluation d'interfaces), notre principale perspective est d'augmenter l'expressivité des règles syntaxiques et sémantiques. Couplé à cela, l'ajout d'un moteur de règles permettra l'ajout et la vérification de règles plus complexes. Ensuite, nous souhaitons capter les préférences visuelles propres aux utilisateurs (choix des flots, choix de coloriage et disposition du graphe, etc.) et faire émerger un *profil visuel utilisateur*, indépendant du jeu de données et inter-applicatif. A une plus grande échelle d'adoption de l'interface, il est imaginable que l'étude quantitative par l'usage puisse remplacer l'évaluation cognitive actuelle, par une évaluation sociale, telle que le propose ManyEyes [14]. Enfin nous souhaitons contribuer à la visualisation de données continues, via l'auto-organisation dynamique de services face aux flux de données qui peuvent varier avec le temps (en termes de fréquence, débit, etc.) et donc nécessiter différents types de traitements au fil de l'eau.

BIBLIOGRAPHIE

1. Chi, E. H. A taxonomy of visualization techniques using the data state reference model. In *INFOVIS*, pages 69–76, 2000.
2. Do, T.-N., and Fekete, J.-D. V4miner pour la fouille de données. *numéro spécial de la revue RIA, Revue d'Intelligence Artificielle*, 2008.
3. Dragicevic, P., and Fekete, J. Input device selection and interaction configuration with ICON. In *Joint proceedings of HCI 2001 and IHM 2001*, pages 543–448, 2001.
4. Fielding, R. T. *Architectural styles and the design of network-based software architectures*. PhD thesis, 2000. Chair-Taylor,, Richard N.
5. http ://developer.yahoo.com/yui/. Yahoo ! Pipes. 2009.
6. http ://javascript.neyric.com/wireit/. WireIt. 2009.
7. http ://pipes.yahoo.com. The Yahoo ! User Interface Library (YUI). 2009.
8. Johnston, W. M., Hanna, J. R. P., and Millar, R. J. Advances in dataflow programming languages. *ACM Comput. Surv.*, 36(1) :1–34, 2004.
9. Mackinlay, J. Automating the design of graphical presentations of relational information. *ACM Trans. Graph.*, 5(2) :110–141, 1986.
10. Scuturici, M. Dataspace API. Technical report, LIRIS, 2009.
11. Vuillemot, R., and Rumpler, B. Mapping visualization on-demand onto a virtual globe : an appealing complement to browser-based navigation. In *HT '08*, pages 249–250, New York, NY, USA, 2008. ACM.
12. Vuillemot, R., and Rumpler, B. MosaiZ : Interactive Image Mosaics. Technical report, LIRIS, 2009.
13. Vuillemot, R., Rumpler, B., and Pinon, J.-M. *Enterprise Information Systems*, chapter Dissection of a Visualization On-Demand Server, pages 348–360. LNBIP. Springer Berlin Heidelberg, Apr. 2009.
14. Wattenberg, M., Kriss, J., and McKeon, M. Manyeyes : a site for visualization at internet scale. *IEEE Transactions on Visualization and Computer Graphics*, 13(6) :1121–1128, 2007.