# Modèle et Architecture Orientée Services pour la Visualisation d'Information Interactive

*Romain Vuillemot*

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

**RESUME**
L'objectif de nos travaux est de favoriser la création et la personnalisation de représentations visuelles interactives. Notre contribution consiste en un modèle et une architecture orientée services qui fournie les représentations visuelles (graphes, nuages de mots clés, etc.) aux applications clientes interactives. Nous avons également introduit une interface de programmation visuelle afin de faciliter la personnalisation et le partage de ces représentations visuelles au sein de communautés d'utilisateurs, afin d'obtenir une évaluation sociale. Dans nos futurs travaux, nous souhaitons faire émerger un profil visuel utilisateur, et appliquer notre approche à la visualisation de données continues.

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

**ABSTRACT**
Our works aim at fostering creation and personalisation of interactive visual representations. Our contribution consist in a service oriented architecture that provides visual representations (graph, tagclouds, etc.) to interactive client applications. We also introduced a visual programming interface those facilitates visual representation personalisation and sharing, among users communuty to obtain a social evaluation. Our future works rely in the emergence of a visual user profile, and in the application of our approach to continuous data visualisation

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

**GENERAL TERMS:** Design.

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

## INTRODUCTION

La diversité des représentations visuelles d'information (telle qu'illustrée sur le site `visualcomplexity`[1]) est très grande et n'a cessé de croitre depuis une décennie. Cependant, ces représentations visuelles nécessitent des compétences pointues en programmation pour être réutilisées dans d'autres contextes, et avec d'autres jeu de données tel que le sien ou un jeu de donnée de référence afin de comparer les contributions entre elles [2].

L'objectif de nos travaux est de faciliter la réutilisation et la personnalisation des représentations visuelles, afin de permettre aux utilisateurs de mieux exploiter le nombre croissant de données qui les entours. Nous pensons que l'utilisateur doit garder son propre cadre applicatif existant et maitrisé (afin de minimiser la surcharge d'apprentissage). Pour cela nous considérons (de l'extérieur) les applications interactives actuelles comme composées d'un modèle, d'une vue et d'un contrôleur (MVC). Notre problématique peut se formuler ainsi : *comment étant donné un contrôleur fixé à l'avance, permettre d'inclure toute vue sur les données (quelle que soit la métaphore : arbre, liste, ..) et être générique à tout modèle de données (dans la limite de la compatibilité entre le type de données et la visualisation) ?*

Notre problématique consiste donc à offrir un cadre *technique* afin de permettre la réutilisation et la compatibilité de briques logicielles existantes (et actuellement liées entre elles) qui forment les représentations visuelles interactive. Voici nos trois principales contributions :

1. *Un modèle générique de traitement des données :* afin de permettre la composition des traitements qui conduisent à la représentation visuelle des données.
2. *Une architecture orientée services :* afin de publier chaque étape traitement (qui composent ensemble une représentation visuelle) sous forme d'une URL.
3. *Intégration dans un cadre applicatif :* afin de coupler et interagir avec les représentations visuelles dans l'application (contrôleur) locale interactive à l'utilisateur.

## MODELE GÉNÉRIQUE DE TRAITEMENT DES DONNÉES

Le *Data State Reference Model* introduit par Ed Chi [1] offre déjà un cadre analytique de référence, pour une meilleure compréhension des différentes étapes de traitement qui mènent à la visualisation de données. Ces étapes

[1] http ://visualcomplexity.com

sont successives, et forment ce que Heer [4] appelle un *pipeline* et qui peut servir de base pour les programmeurs afin d'implémenter des représentations visuelles interactives. Cependant, une fois l'implémentation réalisée, même si les étapes ont bien été identifiées elles restent verrouillées dans l'application, sans qu'il soit possible de les changer sans risque de déclencher un long et incertain nouveau cycle de développement.

Notre approche [10] s'inspire de ce modèle en *pipeline*, et propose de rendre les traitements de données génériques en les découpant et en les publiant sous forme de services Web, classés en différentes catégories : extraction, sélection, disposition et rendu. La recomposition de ces services constitue alors des flots dont l'implémentation et l'environnement d'exécution ne deviennent plus contraignants : les seules contraintes apparaissent à deux niveaux dans 1) l'appariement local des entrées/sorties et 2) le respect de contraintes globales comme la vitesse d'exécution, passage à l'échelle ou cohérence de la visualisation. Dans le premier cas il s'agira de règles *syntaxiques* assez basiques et faciles à implémenter, dans le second cas de règles *sémantiques* plus complexes car même si elles apparaissent dans la littérature, leur formalisation reste un défi ouvert.

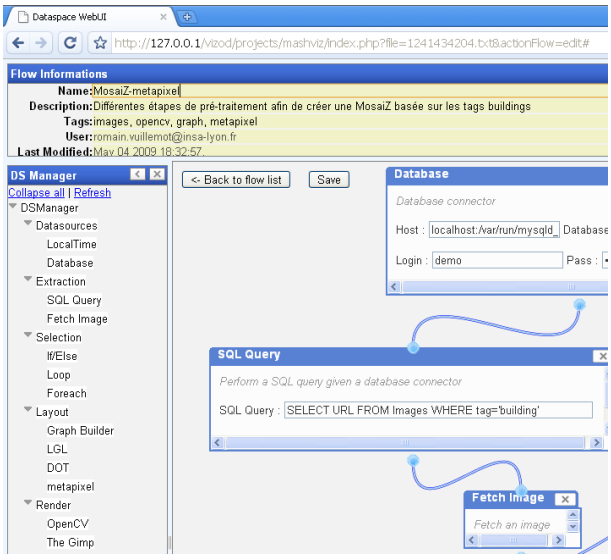


FIG. ***1*** **:** Extrait de l'interface de programmation visuelle qui permet la composition de services web. A gauche l'annuaire de services et à droite le flot créé à partir de ces services.

Dans cette optique, et afin de rendre les compositions de flot accessible aux utilisateurs, nous avons introduit une interface de programmation visuelle [9] (Figure 1). Cette interface type web ne demande pas d'installation préalable et permet de manipuler des services qui sont eux distribués. Ces services sont disponibles à partir d'un annuaire et offre un outil collaboratif aux programmeurs, designers et utilisateurs qui peuvent contribuer à améliorer une visualisation interactive, tout en travaillant de manière asynchrone et en ne se focalisant que sur leur domaine d'expertise. Nous avons également introduit une interface de partage et d'annotation des flots pour la réutilisation d'expériences liées à la composition de ces flots, qu'il est possible de cloner et modifier (afin de ne pas partir de zéro).

## ARCHITECTURE ORIENTÉE SERVICES

Les récents travaux de Fielding [3] ont permis de formaliser un style d'architecture d'accès aux services web, dit REST (Representational State Transfer), et qui les rend disponibles sous forme de ressources accessibles via une URL (et donc non sensibles aux proxies). Le web est le meilleur exemple d'implémentation d'accès à ces ressources. Dans notre cadre, les ressources sont les flots précédemment créés à partir de la composition de traitements, et dont la sortie est la représentation visuelle qui sera lue par le contrôleur (application interactive cliente). La Figure 2 illustre nos propos via l'intégration d'un tagcloud dans notre interface POSvis [7] pour l'exploration et la caractérisation d'entités dans des collections de textes. L'interface est développée en Java Swing et accède à la vue des données par simple commande `GET` sur l'URL, et l'utilisateur peu interactivement changer l'association données et attributs (via des listes de sélection). L'URL qui pointe vers les données est unique, et peut donc facilement être partagée et intégrée dans tout autre environnement interactif (un navigateur web par exemple).

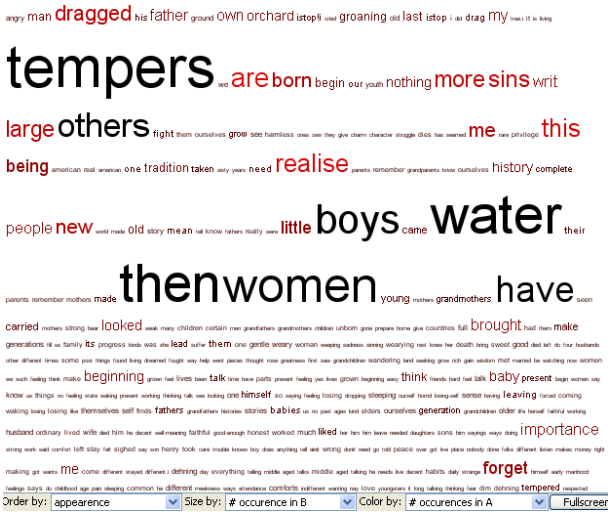


FIG. ***2*** **:** Voici un tagcloud obtenu via une URL, et résultant d'un flot de services. Dans cet exemple l'ordre des mots du tagcloud respecte celui de leur ordre d'apparition dans le corpus A, la couleur en fonction de la fréquence dans ce même corpus, et la taille est encodée en fonction de la fréquence des mots dans un corpus B.

Voici l'URL (simplifiée et commentée) :

```
// Informations sur la source des données
http://HOST/posvis/tagcloud.php?chapter=1&collection=1
// Informations sur la sélection des données
 &startSelectionA=0&endSelectionA=4892&startSelectionB=4893&endSelectionB=6112
// Correspondance données/attributs visuels
 &orderBy=appearence&sizeBy=frequencyInB&colorBy=frequencyInA
// Seuillage pour permettre une mise en avant des mots importants
 &minSize=50&maxSize=500&limitMinSize=50
```

D'autres approches similaires, comme Manyeyes [11] d'IBM, proposent aussi la mise en ligne de la vue sur les données afin de générer des interactions utilisateurs, comme commenter et annoter. Ainsi une nouvelle forme d'évaluation *sociale* émerge, par opposition à l'approche *cognitive* classique. Cependant l'interface interactive proposée reste monolithique en Java et les options (couleurs, etc.) limitées. Enfin les données ne sont pas exportables car elles restent la propriété d'IBM. Il n'est donc pas permis une plus grande diversité d'usages, ni d'évaluation car l'accès aux statistiques des données n'est pas possible. A l'opposé, notre interface de partage et d'annotation de flots [9] ouvre l'accès aux sources et statistiques détaillés d'usages des flots, afin que les programmeurs puissent mieux analyser le fonctionnement de leurs contributions et ensuite itérer à nouveau.

## INTÉGRATION DANS UN CADRE APPLICATIF

Les flots (ou vues sur les données) sont ensuite intégrés dans un cadre applicatif pour que les différentes variables qui les composent puissent être manipulées. Et ainsi permettre à l'utilisateur de converger vers une représentation visuelle optimale, ou simplement assister son raisonnement. Tout type d'environnement interactif peut donc joueur le rôle de contrôleur, avec des widgets adaptés aux données (par exemple un slider pour les variables numériques, ou l'altitude sur un globe terrestre permet la multi-résolution [8], passage de paramètres à une URL en fonction de choix d'éléments dans une liste ou de choix multiple [7], etc.). Afin de coupler nos flots de données avec les applications actuelles, nous avons développé une API [5] qui normalise les appels aux services Web. Cette API peut soit être inclue lors du développement de l'application, soit être à l'interface entre l'application et les requêtes sur la vue de données (par exemple implémenté dans un proxy). Des fonctionnalités supplémentaires comme l'upload du jeu de données, l'identification de l'utilisateur et le traçage de session inter-applicatif ont été introduites.

## DISCUSSIONS

### Profil visuel utilisateur

Un de nos principal objectif est de faire émerger un profil utilisateur *visuel*. Un profil est une façon de caractériser un individu en fonction de ses préférences explicites (choix de personnalisation) et implicites (usages). Contrairement au profil de *données*, le profil *visuel* est indépendant de toute instance de données. Par exemple, pour les données types graphes, nous pouvons capter deux types de choix explicites de personnalisation (indépendants du contenu du graphe), qui pourront faire partie du profil visuel :

***Choix du layout du graphe.*** Cette étape est importante car une des spécificités intrinsèque de la visualisation d'information, est que la représentation des données est choisie plutôt que subie. Nous avons implémenté la bibliothèque Dot sous forme de service web et rendue accessible comme une étape de traitement de flots (Figure 3).

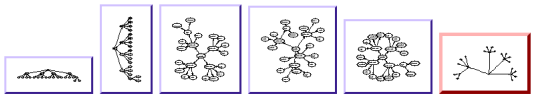

FIG. ***3*** **:** Le choix des layouts (disponibles sous forme de services) est laissé libre à l'utilisateur.

***Choix des associations.*** L'association données et attributs visuels est une étape cruciale qui contribue à la pertinence de la représentation des données. Nous proposons à l'utilisateur de formuler ses règles de coloriage (Figure 4) sous forme d'associations génériques et réutilisables.


FIG. ***4*** **:** Règle de coloriage d'un graphe sous forme d'associations, et disponible sous forme de service.

Par ce moyen, l'utilisateur peut encoder ses réflexes visuels, et se décharger cognitivement afin d'effectuer seulement une détection de formes ou groupes d'intérêts car il a déjà intégré la codification de la représentation.

### Passage à l'échelle

Une autre préoccupation a été de tester la capacité à traiter de grandes quantités de données. Concernant la communication inter-service nous avons fait par exemple des tests concluants avec des traitement d'images atteignant la dizaine de mega-octets. Un léger temps de latence apparaît lors de l'appel aux flots (et dû à la communication réseau) mais qui n'est quasiment pas perceptible. Concernant le traitement de grands volumes de données, l'exécution à distance des services permet de concentrer les moyens techniques et humain sur un serveur, voir aussi d'exécuter les services sur un cloud ou architecture distribuée. Enfin une prochaine étape est d'intégrer une codification visuelle dans notre interface de programmation visuelle afin de facilement détecter les goulots d'étranglement en termes de temps et de débit.

### Limites potentielles

La principale limite réside dans la modélisation de l'utilisateur. Notre API permettant le traçage de l'activité dans les applications, nous avons modélisé l'utilisateur comme un automate où chaque état est l'accès à une vue sur les données, et la transition l'interaction d'une vue à l'autre. Ce modèle nous semble pertinent et nous a déjà permis de visualiser l'activité utilisateur comme sur la Figure 5 sous forme d'un diagramme d'états, et montre comment le mantra de Shneiderman *"Overview first, zoom and filter, then details-on-demand"* [6] peut émerger via la navigation multi-résolution de grands graphes [8]. Le diagramme montre que l'application ne permet pas une vue globale de suite (le sommet rond noir indique le début de session), et que certains utilisateurs passent directement des détails à la vue globale, sans l'étape de zoom.

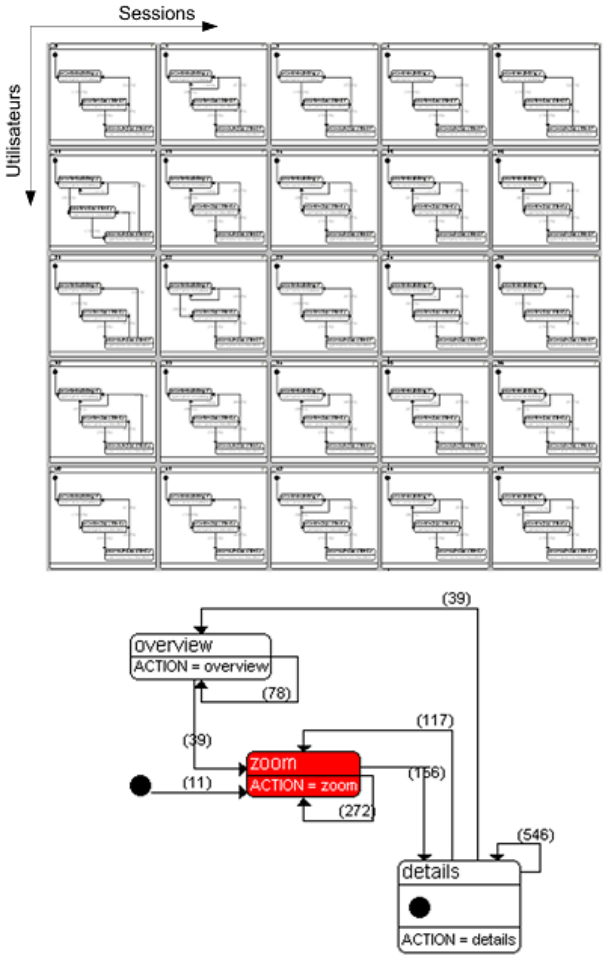


FIG. ***5*** **:** Nous avons capté et modélisé l'activité de l'utilisateur sous forme d'état. La partie du haut montre les sessions et les utilisateurs. La partie du bas montre l'agrégation de ces sessions et utilisateurs, afin de détecter les comportements statistiquement fréquents.

Une segmentation des utilisateurs serait à réaliser afin de mieux analyser les comportements, basée par exemple sur des similarités (fréquence, nombre) d'évènements qui les déclenchent. Une limite qui apparait ici réside dans le fait que l'on ne capte que les appels au serveur et ceux ci ne sont pas permanents, par exemple l'utilisateur peut effectuer des opérations de filtrage locales qui ne seront pas captées. Aussi, capter les appels ne garantie pas d'observer un comportement exact de l'utilisateur, qui peut être distrait ou effectuer une autre tâche.

## TRAVAUX FUTURS

Notre prochains travaux consistent à reprendre et étudier les différents paradigmes existants en visualisation interactive (vues multiples coordonnées, approches par multi-résolution, etc.) et d'essayer de les formaliser sous formes de règles *syntaxiques* et *sémantiques* qui seront implémentées dans l'interface *mashviz*. Un autre domaine qui nous intéresse particulièrement comme piste future de recherche est la visualisation de données continues. Ce type de visualisation offre un défi nouveau, car une incertitude forte réside sur la taille et la fréquence des données. Il sera donc nécessaire de bien caractériser les services afin de garder une interactivité et une consistance de l'interface. Ainsi des études approfondies sur les comportements des flots et leur auto-organisation contrainte permettrait de répondre à ce défi.

## BIBLIOGRAPHIE


1. Chi, E. H. A taxonomy of visualization techniques using the data state reference model. In *INFOVIS*, pages 69–76, 2000.
2. Fekete, J.-D., and Plaisant, C. Les leçons tirées des deux compétitions de visualisation d'information. In *IHM 2004 : Proceedings of the 16th conference on Association Francophone d'Interaction Homme-Machine*, pages 7–12, New York, NY, USA, 2004. ACM.
3. Fielding, R. T. *Architectural styles and the design of network-based software architectures*. PhD thesis, 2000. Chair-Taylor,, Richard N.
4. Heer, J., Card, S. K., and Landay, J. A. prefuse : a toolkit for interactive information visualization. In *CHI*, pages 421–430, 2005.
5. http ://vizod.liris.cnrs.fr/api/. Visualization on-demand API. 2008.
6. Shneiderman, B. The eyes have it : A task by data type taxonomy for information visualizations. In *VL '96 : Proceedings of the 1996 IEEE Symposium on Visual Languages*, page 336, Washington, DC, USA, 1996. IEEE Computer Society.
7. Vuillemot, R., Clement, T., Plaisant, C., and Kumar, A. What's Being Said Near "Martha" ? Exploring Name Entities in Literary Text Collections. In *IEEE Symposium on Visual Analytics Science and Technology (VAST)*, october 2009.
8. 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.
9. Vuillemot, R., and Rumpler, B. Une interface de programmation visuelle pour la composition de services de visualisation d'information. In *IHM'09 : Actes de la 21ème Conférence Francophone sur l'Interaction Homme-Machine*, 2009.
10. 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.
11. 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.