UN CLIENT WEB POUR DÉCOUVRIR, EXPLORER ET TESTER DES TRAITEMENTS SUR DES DONNÉES GÉOGRAPHIQUES

par Bénédicte Bucher

Laboratoire COGIT
Institut géographique national
2 -4 avenue Pasteur, 94165 Saint-Mandé cedex
benedicte.bucher@ign.fr

1 Introduction

La modularité des composants manipulant des données géographiques s'est largement accrue ces vingt dernières années. Les spécifications standard d'interfaces entre composants logiciels géographiques doivent permettre le développement de modules spécifiques 'interchangeables'. Ces modules sont par exemple des services Web ou des bibliothèques java téléchargeables sur le Web. Ainsi, les services de généralisation MapShaper et WebGen permettent à tous d'utiliser des traitements de généralisation [1] [2]. Des exemples de bibliothèques java proposant des traitements de données géographiques sont Geotools et Geoxygene. Elles implémentent un même modèle conceptuel, celui d'ISO/OGC. Dans ce contexte, il est important que tout utilisateur potentiel d'un module puisse connaître son existence, évaluer son intérêt et savoir l'utiliser [1] [2].

Cet article présente un travail en cours pour définir une interface Web permettant de découvrir et explorer des traitements géographiques disponibles sous forme de services Web invocables à distance ou sous forme de modules interopérables téléchargeables sur le Web.

L'objectif immédiat est de faciliter la conception d'une application qui utilise ces traitements. Nous voulons permettre à un utilisateur d'évaluer les implémentations disponibles de fonctionnalités afin d'en choisir une et de l'intégrer dans son programme, que ce soit sous la forme d'un appel distant à méthode (si cette fonctionnalité est fournie par un service Web) ou sous la forme d'un appel local de méthode (si cette fonctionnalité est fournie par une bibliothèque).

L'objectif secondaire est d'encourager les auteurs de bibliothèques java interopérables à identifier et décrire des méthodes de leurs bibliothèques, qui sont pertinentes en dehors de leur bibliothèque, c'est-à-dire à améliorer l'interface programmatique (et sa documentation) de ces bibliothèques. Il existe des environnements permettant de déployer des méthodes java sous la forme de services Web tel axis. Ces environnements gèrent les aspects de communication réseau –comprenant la sérialisation XML par exemple— et l'écriture de métadonnées standard. Avant cela, le développeur doit identifier les méthodes pertinentes, éventuellement renommer la méthode ou les types pour en faciliter la compréhension par l'utilisateur, améliorer le corps de la méthode pour la rendre plus robuste ou renseigner les conditions d'utilisation de cette méthode. Il doit aussi prévoir des messages d'erreur pertinents en cas d'échec d'exécution de la méthode. Il nous semble que ce travail sera d'autant plus gratifiant s'il se fait en collaboration avec des utilisateurs de la méthode, ce que doit permettre cette application couplée à un éditeur de métadonnées.

L'auteur décrit sa méthode grâce à un éditeur de métadonnées. Des utilisateurs découvrent et testent sa méthode grâce à l'application qui exploite les métadonnées. Le modèle de métadonnées que nous utilisons a été proposé en reprenant des rubriques de modèles dédiés à la description fine de services Web [4]. L'éditeur de métadonnées n'est pas présenté dans cet article.

Dans la suite de cet article, nous adaptons des principes provenant des infrastructures de données spatiales [5] et de la recherche d'information et les adaptons à notre contexte : l'accès d'utilisateurs à des traitements sur des données géographiques fournis par divers composants logiciels. Cela nous conduit à distinguer trois phases d'accès illustrées sur la figure 1 : découverte, exploration et test. Pour

chacune de ces phases, une analyse fonctionnelle est suivie par une description de notre proposition.

Le prototype en cours d'implémentation est une applet java signée qui communique avec des servlets.

2 Découvrir des traitements potentiellement intéressants

La phase de découverte vise à identifier, parmi un ensemble de ressources souvent nombreuses, un sous-ensemble de ressources potentiellement pertinentes en réponse à une requête d'utilisateur. Deux fonctions importantes sont :

- supporter l'expression de besoin de l'utilisateur ;
- sélectionner des ressources pertinentes en réponse au besoin exprimé.

Un moteur de découverte s'appuie sur des données qui sont les ressources elles-mêmes ou des données à propos de ces ressources (par exemple des index). De telles données dédiées à la découverte doivent comporter des critères pertinents pour l'utilisateur (par exemple les mots contenus dans un document Web ou l'extension spatiale d'un jeu de données géographiques). Ces critères doivent être valables pour toutes les ressources (par exemple le critère spatial ne convient pas pour chercher tous les documents du Web car certains n'ont pas d'aspect spatial pertinent pour l'utilisateur). Ces données doivent être organisées dans une structure requêtable et elles doivent indexer les ressources.

Dans notre contexte, un type de ressource intéressant est celui des services. Le W3C définit un service comme 'une ressource abstraite qui représente la capacité de réaliser une tâche correspondant à une fonctionnalité cohérente' [3]. Le composant logiciel réalisant effectivement cette fonctionnalité est appelé agent. Nous souhaitons utiliser la notion de service pour indexer les composants logiciels qui fournissent des traitements, comme illustré sur la figure 2 : l'utilisateur identifie d'abord le service qui l'intéresse et ce service pointe in fine vers un composant logiciel qui le fournit.

Dans la suite, nous nous intéresserons donc à la littérature sur la description de services en vue de faciliter l'identification d'un service d'intérêt. Un modèle important est le modèle de métadonnées pour cataloguer les services Web UDDI [6]. Les critères de découverte d'UDDI sont les pages blanches (qui fournissent le service) et les pages jaunes (ce que fait le service). Ces métadonnées sont remplies en faisant référence à des taxonomies. En effet, dans le cas de la phase de découverte, l'usage d'un vocabulaire contrôlé est essentiel. La dernière catégorie de métadonnées est ce qu'on appelle les pages vertes (comment utiliser le service). Elle correspond au référencement des ressources indexées. UDDI est repris schématiquement sur la figure 3.

Nous nous intéressons maintenant aux taxonomies permettant de décrire des services abstraits (les pages jaunes). La taxonomie de services ISO19 119 identifie des catégories de très haut niveau de services géographiques parmi lesquelles 'geographic processing services'. Cette dernière catégorie est divisée en quatre sous-catégories selon la nature de l'information traitée : spatiale, thématique, temporelle, métadonnée. Chaque sous-catégorie est elle-même spécialisée. Cette taxonomie n'est pas exhaustive et est trop générique. Par exemple [2] propose des classifications plus spécifiques pour des services de généralisation (catégorie qui apparaît dans la taxonomie ISO19 119, mais qui ne comporte pas de sous-catégorie). Une de leurs classifications s'appuie sur les types de données sous-jacents, en supposant que les services utilisent tous GML comme format de données géographiques. La catégorie 'Default generalisation services' comprend ainsi les services dont les entrées et sorties s'expriment à l'aide des éléments de GML simple. Les autres services appartiennent à la catégorie 'Advanced generalisation services'. Une autre des classifications de [2] s'appuie sur les phases de généralisation. Sont identifiés les 'generalisation support services' qui préparent les données, les 'generalisation operator services' qui transforment les données, les 'generalisation process services' qui contrôlent le processus global de généralisation. Cette classification montre l'importance d'une autre relation que la subsumption entre les catégories d'une classification de services : la chronologie (ce service doit être utilisé avant celui-là) et la composition (ce service utilise celui-là). Par ailleurs, un autre challenge, non abordé par les auteurs, est de décrire des services spécifiques de façon non ambiguë et d'exprimer de façon formelle les relations entre services. [7] se penche sur ces aspects et propose une ontologie d'opérations, OPERA, écrite en OWL. Des travaux plus anciens ont pris place dans ce contexte

nommer et modéliser les manipulations de données géographiques dans un SIG. [8] et [9] ont formalisé des opérations abstraites d'analyse spatiale dans leur 'Map Algebra'. Plus tard, [10] a proposé une liste d'opérations abstraites universelles correspondant aux fonctions SIG. Son système Virtual GIS permet à un utilisateur de spécifier à un niveau abstrait des séquences d'opération et des les invoquer sur ses données. [11] a ajouté à VGIS des opérations hybrides sur les données vecteur et raster. D'autres auteurs se sont concentrés sur des services abstraits de plus haut niveau, comme la cartographie de données thématiques [12]. Une telle expertise devrait être intégrée dans un modèle de services abstraits pour faciliter la découverte de logiciels basée sur des objectifs d'utilisateurs [13].

Pour résumer, il existe plusieurs classifications ou autres modèles pour décrire des services abstraits allant d'opérations élémentaires à des processus complexes. Un modèle de découverte de services pertinents pour un utilisateur doit intégrer des catégories provenant de ces modèles dans un modèle unifié. Des critères importants pour la découverte sont le nom du service, sa signature, les relations de subsumption et de composition avec d'autres services. Nous proposons un modèle ad hoc résumé sur la figure 4. Dans ce modèle, le terme 'fonction' est employé au lieu de celui de 'service'. La formalisation des relations de composition (stratégies et méthodes) s'appuie sur des diagrammes d'activité [14]. Une fonction est associée à une activité qui décrit comment la réaliser. Cette activité peut consister en une séquence d'échanges de messages XML avec un service Web ou d'appels à méthodes.

Dans le prototype, l'utilisateur peut parcourir les instances de ce modèle présentées dans une structure d'arbre. Un nœud fonction a plusieurs fils : ses variables, des fonctions plus spécifiques, des composants logiciels et des activités.

3 Explorer les ressources trouvées

L'exploration vise à préciser la pertinence des ressources sélectionnées lors de la découverte, et ce en amont de leur utilisation, c'est-à-dire seulement à l'aide de leurs métadonnées. Une fonction importante est l'identification de la ressource décrite (par exemple les énoncés des URL et des titres associés dans la réponse d'un moteur de recherche). Une autre fonction importante est l'affinement de l'évaluation de la pertinence. Cela peut se faire en proposant à l'utilisateur des métadonnées interprétables par lui (comme les fragments du document entourant les mots qu'il a utilisés pour sa requête). Une dernière fonction est de comparer les pertinences des ressources. Un moteur de recherche peut ainsi utiliser des critères (popularité d'un site, fréquence du mot-clé) et des algorithmes complexes pour ordonner et regrouper les documents présentés en réponse à une requête.

3. 1 Identification de fonctions basées sur des illustrations cartographiques

Dans notre contexte, l'utilisateur doit identifier la fonction (le service abstrait) puis le composant logiciel. L'identification des composants repose sur les URLs des services Web et sur les noms java complet des bibliothèques. L'identification des fonctions est un problème complexe car il n'existe pas d'ontologie universelle de fonctions géographiques. Pour autant, quand la fonction est cartographique ou modifie la géométrie, un aperçu visuel peut faciliter son identification par l'utilisateur. Dans [4], le développeur d'une méthode peut l'illustrer en l'appliquant à des données test hébergées sur le catalogue. Une illustration plus fine peut être une mosaïque de résultats correspondant à différentes valeurs de paramètres et différents types de données en entrée. En effet, certains traitements géographiques complexes se comportent de façons très différentes en fonction des valeurs des paramètres et des propriétés des données (taille des objets, sinuosité des routes, alignement de bâtiments …). Ce problème a été étudié par [15]. Les auteurs ont construit une base d'échantillons cartographiques obtenus en appliquant un même algorithme de généralisation avec différentes valeurs de paramètres à différents types de données géographiques.

3. 2 Détermination de la pertinence basée sur les activités

La pertinence d'un composant pour fournir un service doit inclure l'efficacité et l'efficience. Nous nous concentrons sur l'efficacité. Rappelons que dans notre modèle illustré sur la figure 4, des activités décrivent comment réaliser une fonction à l'aide de composants logiciels indexés. L'efficacité d'un composant pour qu'un utilisateur réalise une fonction sera grandement liée à des propriétés de l'activité correspondante et au profil de l'utilisateur. Les propriétés à prendre en compte sont : le nombre d'ac-

tions, le nombre de composants logiciels additionnels, l'existence de valeurs par défaut pour les variables, la complexité des types de données en entrée et en sortie.

4 Tester les ressources sélectionnées

Enfin, l'utilisateur a besoin de tester les ressources sur ses données, c'est-à-dire de réaliser la fonction choisie sur ses données. Cela nécessite de télécharger ses données sur le serveur, de remplir les valeurs des variables de l'activité sélectionnée et de demander au serveur d'exécuter l'activité. Ensuite, l'utilisateur doit 'voir' le résultat. Il peut aussi vouloir essayer d'autres valeurs de paramètres, tester des séquences, revenir en arrière et recommencer. Nous décrivons ci-dessous comment l'utilisateur remplit les valeurs des variables d'une activité.

4. 1 Spécifier les valeurs des variables de l'activité sélectionnée

La spécification des variables d'une activité se fait grâce à un composant générique appelé 'obrowser' (fig.5) Obrowser permet via une interface graphique d'éditer directement les champs publics d'un objet et d'invoquer n'importe quelle méthode java. Cette généricité est importante car les domaines de valeur des variables peuvent correspondre à n'importe quelle structure. Obrowser peut aussi être étendu pour afficher de façon spécifique les objets de certaines classes. En particulier, il a été étendu pour afficher différemment les activités –par exemple pour permettre à l'utilisateur de se concentrer sur les champs qui ont un sens pour lui. Nous supposons donc que toute valeur de variable peut s'écrire sous forme d'un objet java. Si en pratique, une valeur est un fragment XML, il est possible d'utiliser une traduction en java de la structure de ce fragment.

4.2 Distribuer spécification et exécution entre le client et le serveur

Lorsque l'utilisateur a suffisamment spécifié une activité, celle-ci devient exécutable et il peut demander au serveur de l'exécuter. Le serveur doit lui renvoyer une vue du résultat en lui signalant que c'est le résultat de cette activité spécifiée. Par ailleurs, l'utilisateur doit pouvoir revenir en arrière pour comparer visuellement plusieurs jeux de données ou pour essayer une étape suivante.

L'objet 'HistoriqueSession' stocke, sous la forme d'une activité en cours de construction, toutes les spécifications et demandes d'exécution d'activités faites par l'utilisateur ainsi que les résultats des exécutions conduites par le serveur à la demande du client. Il permet au client et au serveur de synchroniser leurs connaissances des spécifications et exécutions. Il permet à l'utilisateur de revenir à une étape précédente. Cet objet est connu du client et du serveur sous deux versions différentes. La version du serveur comprend les valeurs véritables de toutes les variables des activités, y compris celles qui sont des données géographiques. La version du client ne comprend pas les valeurs des variables qui sont des jeux. La description d'une activité manipulant des données géographiques sur le client utilise, au niveau de ces variables, des données SVG qui sont la traduction des FeatureCollections hébergées sur le serveur.

Ainsi, lorsque l'utilisateur télécharge ses données sur le serveur, celui-ci génère un document SVG et le renvoie au client avec un identifiant. Le serveur met à jour sa version de l'HistoriqueSession en ajoutant une étape, qui peut être retranscrite en langage naturel de la façon suivante: chargement du jeu de données identifié par NoX, NoX correspond à telle FeatureCollection. Puis il envoie au client ce que ce dernier doit ajouter à sa version de l'Historique Session : chargement du jeu de données identifié par NoX, NoX qui correspond à tel document SVG. La différence entre les deux HistoriqueSessions est donc que les identifiants renvoyant à des Feature Collections chez l'un renvoient à des documents SVG chez l'autre.

Par ailleurs, lorsque l'utilisateur travaille sur un jeu de taille importante, le serveur ne travaille pas sur tout le jeu, mais seulement sur un extrait qui est généré automatiquement lors du chargement.

4.3 La cartographie SVG du résultat d'une activité

La visualisation du résultat d'une activité exécutée est basée sur deux éléments. Le premier est une bibliothèque de styles SVG dédiés au dessin des effets d'un traitement. Ces styles sont limités à certains effets du type 'modification d'un objet' : objet détruit, objet créé, attributs modifiés. Le style objet créé est spécifié en classe créée et en géométrie créée. Le serveur calcule les classes d'objets correspondantes, c'est-à-dire l'ensemble des objets créés, etc. Par exemple, le style 'géométrie créée' est un style adapté à l'illustration des buffers.

Le deuxième élément est un ensemble de types de mosaïques de cartes attachés à différents types d'éléments de l'HistoriqueSession. Chaque activité de l'HistoriqueSession a un élément 'décomposition' qui mène aux sous-activités composant celle-ci. Cet élément est associé à une mosaïque de cartes qui correspondent aux résultats de toutes ces sous-activités. Cette mosaïque peut être affichée sous forme chronologique ou sous forme workflow comme illustré sur la figure 6.

Tout élément 'Variable' de l'HistoriqueSession est attaché à un ensemble de cartes SVG correspondant aux résultats des activités suivantes : activités ayant produit la valeur de cette variable et activités qui consomment la valeur de cette variable.

Conclusion

Ce papier a présenté une proposition en cours d'élaboration et de prototypage pour faciliter la découverte et l'exploration sur le Web de traitements géographiques. Cette proposition repose sur un modèle de métadonnées qui décrit des ressources logicielles par le biais des fonctions (services) qu'elles proposent et par les activités correspondantes (qui permettent d'obtenir un service à partir d'une ressource). Cette proposition repose aussi sur un modèle (l'HistoriqueSession) pour synchroniser simplement la vue du client et celle du serveur sur les activités en cours de spécification, les activités exécutées et leurs résultats. Et enfin, elle repose sur des vues cartographiques spécifiques pour explorer le résultat d'une ou plusieurs activités.

Bibliographie

Harrower M., Bloch M., 2006, "MapShaper.org: A Map Generalization Web Service", *IEEE Computer Graphics and Applications*, vol 26(4), p.22-27.

Burghardt D., Neun M., Weibel R., 2005, "Generalization Services on the Web - A Classification and an Initial Prototype Implementation", *CaGIS*, vol. 32, n° 4.

W3C working group, Web Services Architecture, W3C Note, 2004.

Abd El Kader Y., Bucher B., 2006, "Cataloguing GI Functions provided by Non Web Services Software resources Within IGN", dans *AGILE conference*, Visegrad, 2006 (apendum).

Nerbert Douglas D., ed., 2004, *GSDI, Developing Spatial Data Infrastructures* : *The SDI Cookbook*, v2.0.

UDDI Spec Tec, UDDI Version 3.0.2, 2004.

Lemmens R., 2006, *Semantic interoperability in distributed geo-service*, PhD thesis, ITC, Enschede.

Berry Joseph, 1987, "Fundamental Operations in Computer-Assisted Map Analysis", *IJGIS*, vol 1, n° 2, p. 119-136.

Tomlin C. D., 1991, "Cartographic Modelling", in *Geographical Information Systems : Principles and Application*, ed. by D. J. Maguire, M. F. Goodchild and D. Rhind, Harlows, Longmans, vol. 1, p.361-374.

Albrecht J., 1996, "Universal GIS operations for environmental modelling", dans *Proceedings of the 3rd International Conference on Integrating GIS and Environmental Modeling,* Santa Barbara.

Jung S., Voser S. A., Ehlers M., 1998, "Hybrid Spatial Analysis Operations as a Foundation for Integrated GIS", dans *Proceedings of the ISPRS Commission IV Symposium*, Stuttgart, Germany.

Andy Mitchell, 1999, *The ESRI Guide to GIS Analysis,* volume 1 : Geographic patterns & relationships, USA.

Bucher B., 2003, "Translating user needs for geographic information into metadata queries", dans *Proceedings of the 6th AGILE conference*, Lyon, p. 567-576.

Bucher B., Balley S., Richard D., Cébelieu G., Hangouët J-F., 2005, "Shareable descriptions of data production processes", dans *Proceedings of the 8th AGILE conference*, Estoril, Portugal.

Hubert F., Ruas A., 2003, *A method based on samples to capture user needs for generalisation*, 5th Workshop on Progress in Automated Map Generalization, Paris.

Figure 1 : Phases dans l'accès d'un utilisateur à des traitements sur des données géographiques fournis par des composants logiciels divers

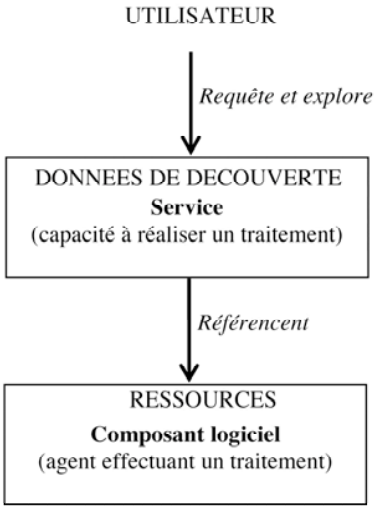

Figure 2 : La description des services (que nous appellerons fonctions) rendus par les ressources qui nous intéressent constituent un bon niveau pour la découverte de ces ressources

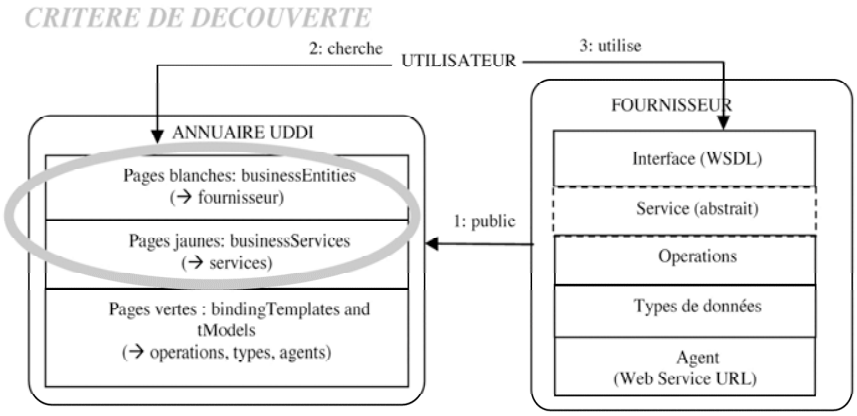

Figure 3 : Fonctionnement d'un annuaire UDDI

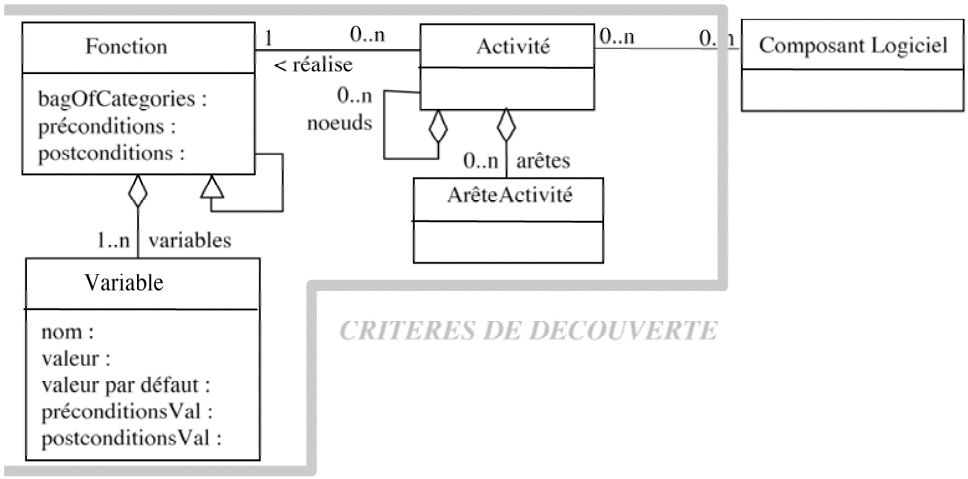

Figure 4 : Grandes lignes de notre modèle pour indexer des composants logiciels pour les fonctionnalités qu'ils fournissent

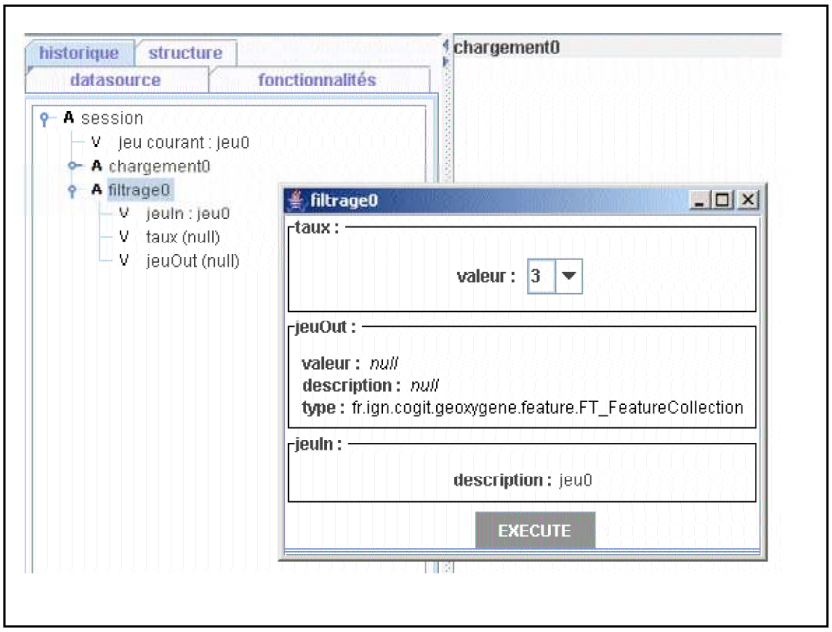

Figure 5 : Utilisation d'Obrowser pour spécifier les valeurs d'une activité spécifique

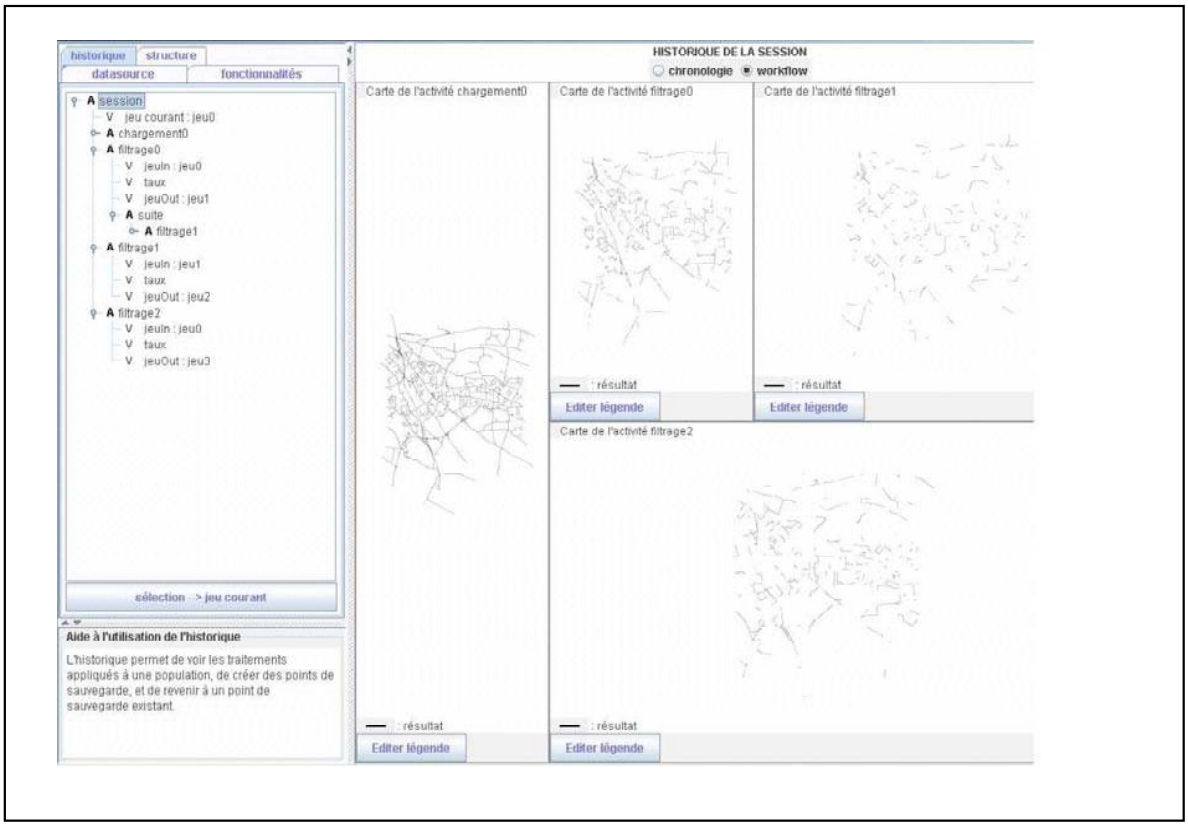

Figure 6 : Affichage de type 'workflow' d'une session. L'utilisateur a chargé ses données puis a appliqué successivement deux filtrages (les deux cartes du haut à droite). Il a ensuite repris les données chargées et a appliqué un seul filtrage avec un paramètre plus fort (carte en bas à droite)