Acquiring service oriented descriptions of GI processing software from experts

Bénédicte Bucher, Laurence Jolivet

COGIT Laboratory, IGN, 2 avenue Pasteur, 94 165 Saint Mandé, France

INTRODUCTION

Metadata are a key item for the diffusion of geospatial processing services. Current works mainly focus on the definition of metadata models to support geospatial processing services discovery, use and chaining. Along with other authors, (Lemmens, 2006) argues for the definition of more detailed metadata models than those stemming from eBusiness. The Open Geospatial Consortium (OGC) has issued a specification for Web Processing Services (WPS, OGC, 2007 a). WPS profiles are to be defined in the 'Geo Processing Workflow' (GPW) thread of the fifth OGC Web Services Initiative (OGC, 2007 b). A complex task, besides defining metadata models and profiles, is the acquisition of metadata. This paper focuses on this last aspect.

In the domain of metadata about data, acquisition is still a crucial issue (Craglia, 2007). We think that the acquisition of services metadata will also present some difficulties, even if these will not be the same. Typically, it is difficult to explicitly define all preconditions and postconditions of a service, as well as its effects and the description of errors.

It is not too early to consider metadata acquisition even if there is no definite standard metadata model and profiles. Indeed, information content that can be found altogether in the most famous Web Services metadata models like WSDL, OWL-S, WSMO and cataloguing models like UDDI remain the same.

We concentrate on involving particular categories of experts in the acquisition of information content that will feed service metadata. Studied categories are people who develop the initial geospatial processing software that underlies the service and people who use this initial software. They may know better the algorithms and the relevant processes (user tasks) than people involved in setting up Web Services architectures. This is why they can contribute in defining accurate descriptions of geospatial processing services based on their algorithms and processes even if they are not familiar with models like WSDL or even with the XML language. To do so, they should not have to learn Web Service technology but they should get familiar with the information content required to diffuse services based on their software. This work aims at assisting developers and users of geospatial algorithms and processes in authoring 'service-oriented descriptions' about the algorithms and processes. These descriptions will be further used to feed metadata for geospatial services based on them.

We firstly detail issues related to this approach. It is especially difficult to relate experts domains of knowledge with concepts specific to Service Oriented Architectures, like that of 'service'. Then we present our current proposal.

OBSTACLES ACQUIRING 'SERVICE DESCRIPTIONS' FROM EXPERTS

This section details obstacles we encountered when we try to let experts (developers or users) formalize descriptive data about their algorithms and processes.

Relating the concept of service to experts knowledge

A first issue is to have experts understand what types of resources we want them to describe. The Service-Oriented Architecture paradigm comes with a new model where the core resource is the service: ‘‘an abstract resource that represents a capability of performing tasks that represents a coherent functionality” from the point of view of a provider or a requester (W3C, 2004). Developers of processing libraries seldom think ‘service’. They often think ‘object’ because they use an object-oriented programming language and ‘algorithms’ if they work in a much procedural domain like image-recognition or generalisation. Users seldom think ‘service’ either. If they use programmatic interfaces, they are familiar with the same concepts as developers –plus packages structures-. If they use graphical interfaces, they rather think in terms of menu items which are not always services. On a basic GIS interface, menu items are not always services. For instance, first menu group is often called ‘File’. Developers and users may use software engineering methods, like these based on UML for instance. In that case, they will be familiar with the notion of ‘capacity’ that is rather close to the one of service.

Clarifying this notion of service revealed a difficult task. A small experiment was performed in the context of a PhD work at COGIT laboratory. The PhD work aimed at designing a catalogue for processes supported by software available within the laboratory (Abd El Kader, 2005). The experiment consisted in asking people what questions they would like to ask to a ‘process’ search engine (the French word we used was ‘traitement’ which means either ‘performing a task’ or ‘the thing that performs a task’). These questions were very heterogeneous in their structure. We translate some of them below.

Which processes have led to this specific feature instance?

What can I use to improve the graphical rendering of my map?

What are the semiologic capacities of the various GIS software available at COGIT lab?

Which operations are used in this specific application?

In which production units are used data matching algorithms?

How do I use this application?

What is the theoretical grounding behind this program?

Which GIS is the best adapted to work on risk data?

I am looking for a program that compares altitudes.

Are there some GIS that support operations jointly on network data and land occupation data?

Are there path analysis algorithms that do not consider shortest paths and what are their parameters?

Who has developed measures about the intrinsic orientation of an object?

Later on, brain storming meetings were organized mainly with developers to formalize the notions of services and processes.

Eventually, the notion of service is hard to identify because it lies between software and usage. Firstly, it is not only associated to the usages of software but also to the software: someone may consider that the relevant task performed by a software component is to compress the data whereas someone else will associate the same software with the relevant task ‘to filter small roads’. Does it mean there are several services here? We think there is just one service ‘generalise roads’ and that it may be associated with the goals ‘to compress data’ or ‘to change data resolutions’. Second, this notion of service is still somehow related to usage. For example, identifying relevant services that are provided by a software component is not a trivial task. The authors experimented this: they designed web services based on an existing application to automatically enhance colour contrasts [Jolivet et al. 07]. This application was to be used following a given process: load data, specify styles, and so on. Identifying, among the steps of this process, functions that may be relevant on their own was possible only thanks to a discussion between the application designer (developer) and potential users of such functions. More generally, developers are often not aware of relevant services that are potentially available based on their code. To put it simply, they do not always think of tasks that will make sense to other users and that are supported by method calls within their code.

These experiments also highlighted the lack of unambiguous vocabulary and natural model to relate the concept of service to usages and to software. Concepts of task, process, operations, algorithms, measures, functions and methods are sometimes fuzzy and so are the relationships between them.

Understanding and evaluating description fields

It is also necessary to explain to developers and users what description items are required. Another ‘user test’ has been performed. We prepared a description form which fields were extrapolated from a compilation of existing models to describe and discover Web Services. This fields were: the service categories (a service may belong to several categories in different classifications), the performed function, the input, the output, preconditions, postconditions, effects, errors that may occur, the possible sequences of interactions with the service, the inner process carried out by the service (its decomposition) and the goals. A description might use several levels of abstraction. For example, an input may be a string pattern encoded after a specific XML structure and, at a higher abstraction level, a postal address the user is going to. In our form, fields were written in natural language. These forms were handled to developers to describe a specific ‘services’ they felt expert about.

The resulting descriptions were rich but very heterogeneous. For instance, a specific kind of service category was the service ‘domain’. This field was documented with ‘Risk analysis’, ‘Generalisation’ or ‘Artificial Intelligence’. ‘Risk analysis’ and ‘Generalisation’ refer to an application domain, ‘artificial intelligence’ to a technical domain. The field ‘function’ was often documented ambiguously. The boundaries between the functions ‘filtering’, ‘selecting’, and ‘querying’ are not very clear. The same problem was raised with the functions ‘acquiring’, ‘deriving’, and ‘extracting’. Preconditions and postconditions have not been understood very well. People found redundancies between fields like the goals, effects, preconditions and postconditions.

Finally some description fields sometimes revealed difficult to value, even when their meaning was unambiguous.

One difficulty consists in assessing the service behaviour accurately enough to document the corresponding description fields: service category, service description, service effects, service postconditions. The same software may have different behaviours depending on the processed data and on the parameters. For example, a building generalisation method will have different behaviours depending on properties of its input (like building size and granularity) and on parameters values. If input data are buildings with non regular shapes, the same processing software will yield squared buildings if the ‘granularity parameter’ is set at a low value and it will yield buildings with enlarged details if the parameter is set at a high value. The general function is the same: ‘to generalise buildings’. However the postconditions are not the same. In one case, buildings have been squared and, in the other one, buildings details have been preserved. This has been explored in (Hubert 2002). He built an application to assist users in valuing parameters of a specific generalisation algorithm according to the behaviour they favoured (among all behaviours of the same algorithm) and according to properties of their input data (in that case, objects size and granularity). Typically, important properties of an input dataset are the spatial properties of real world phenomena represented in the data, like alignments, as defined in (Neun, 2006). Identifying classes of behaviours, identifying relevant properties and the threshold values that correspond to a change of behaviour is a complex task.

Another difficulty is to acquire the requirements of processing software regarding the structure of its input (Bucher, 2007). The developer has to make assumptions about the structure of the representation of the geographic space the program will read and the assumptions are not always rendered in the input datatype definition. The developer may not even be aware of assumptions since 'unit testing' is not possible in our domain. For instance, he may test his network processing software only on datasets with connected arcs with no cycle so that he will not be aware that an error will occur if there is a cycle in the input dataset.

PROPOSED AUTHORING MODEL AND INTERFACE

This section details our current proposal to let people express service-oriented descriptive information about the software they develop or use. First item is a model that explicitly defines concepts at the same time meaningful to these experts and relevant in a service-oriented architecture. Second item is the application developed to let experts edit metadata after this model. So far we have concentrated on the first category of experts: the developers. The same interface will be used to let the second category of experts, the users, refine these metadata.

The model

An ad hoc model has been designed after a review of existing models to describe web services and existing models to describe software (Bucher, 2005). This model focuses on concepts that are meaningful to experts and are relevant in our context. It is also very simple with a limited number of classes. We did not introduce elements to support complex synchronisation. It is schematised on ***Figure 1.*** Part of it has already been described in other papers like (Bucher, 2007).

A key concept coming from service-oriented paradigm is the service profile. We call it *Function*. *Function* parameters are described through a specific class: *Variable*. The set of possible values for a *Variable* is described with several fields of a *Variable* object, not all of them are rendered on ***Figure 1***. One field is *simpleDataType* and it is used to specify an abstract data type for the value –which may be a literal type, or a java class or an xml schema-. Another field is *complexDataType*. It is used to describe the structure of the value when a java class or XML schema is not enough. This field is documented with a *Structure* instance. The *Structure* model is taken from (Balley, 2006) who proposed this model to describe dataset structures in a schema transformation application. A *Structure* contains different interrelated levels of abstraction: the ontological level, the conceptual schema and the logical schema. The latest level is the level of XML schema and java class. The *Structure* model extends ISO/OGC standards to explicitly define the binding knowledge between the different abstraction levels. A *Structure* may be abstract and have no logical schema or it may be purely implemented with no related interpretation. This is important so that people can describe purely abstract *Functions*.

Another important concept is the software component. This concept is very important for developers because it designates something they know. In the context of Web Services this concept would describe the server code (for instance altogether a Web server like Apache Tomcat, the axis application, a java method and a deployment configuration file) but the only relevant aspect would be the endpoint (i.e. the URL where to send the service request).

The relationship between function and software component is not easy to formalise in a meaningful way. We propose to express it through the concept of *Activity*. The *Activity* describes how to carry out a *Function* based on interactions with existing *SoftwareComponents*. An *Activity* describes workflow knowledge as it is usually represented UML2 activity diagrams. Ideally, elementary Activities (i.e. basic interactions with software) should be limited to programmatic interactions. Some specific *Activities* have been modelled to describe the use of a programmatic interface: *CallJavaMethod*, *CallWebService*. We have left the model open to describe also other kind of elementary *Activities* like an interaction with a graphical user interface or any other *Activity*.

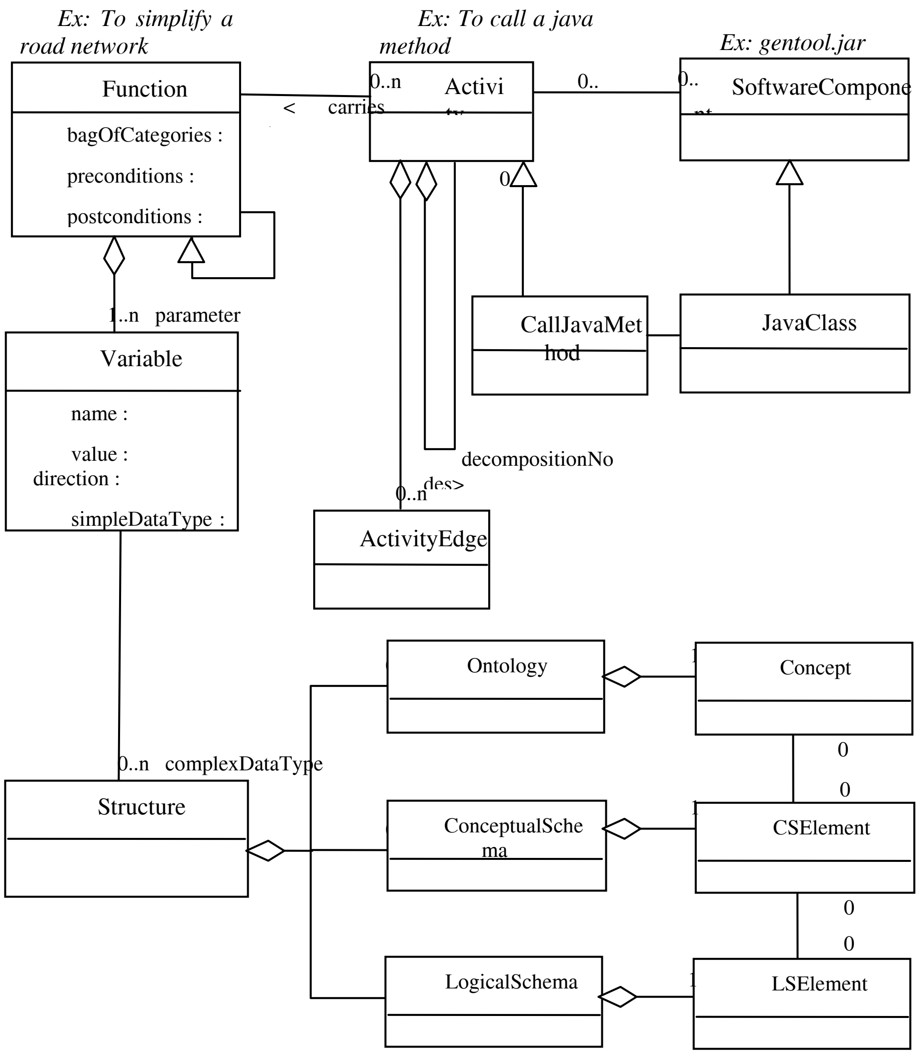
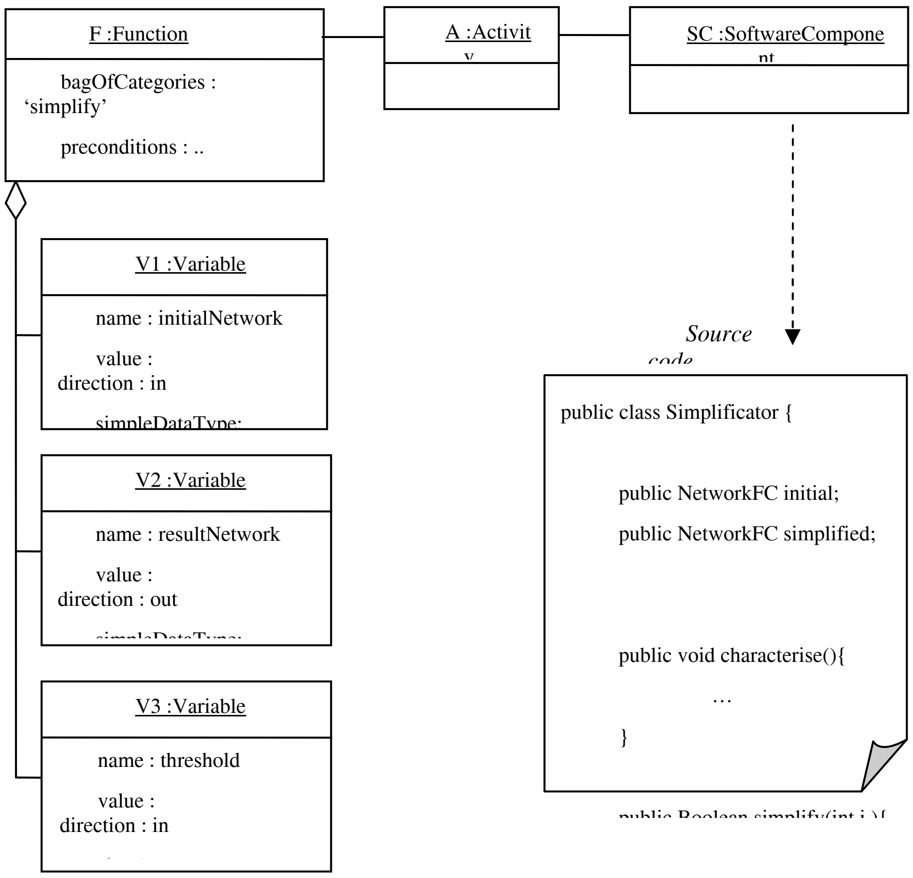


Figure 1. Our model to edit service-oriented metadata about existing processing software and related tasks.

The aim of this model is to support the description of meaningful tasks supported by existing software (the functions). So far, functions (i.e. capacities to perform tasks) are not always explicitly mentioned in software documentation. A simplistic example is illustrated on the ***Figure 2*** and ***Figure 3***.



```
public class Simplificator {

    public NetworkFC initial;
    public NetworkFC simplified;

    public void characterise(){
        …
    }

    public Boolean simplify(int i ){
```

__Figure 2.__ A simplistic example of our service-oriented metadata for a java class. The Activity contains two kinds of knowledge: the sequence of java method calls and the mapping between the Function variables and java objects and fields.

This example on ***Figure 2*** and ***Figure 3*** is detailed hereafter. Simon is a researcher in generalisation who has developed a road network simplification algorithm. His software is a java class called *Simplificator*. The important thing here is to understand that the java method 'simplify' of this java class is not a service at all despite its name. First of all, the processed data do not appear in the method argument and return type (because they are fields of the Simplificator instance). Only one input is described, the simplification threshold. There is as well only one output: a Boolean stating if the input Network was actually simplified or remained unchanged. Simon knows that for the 'simplify' java method to perform well, the initial NetworkFC object should be non null and some of its fields should be documented (like the cycles). This is why Simon always calls the method 'characterise' to document these properties before calling the method 'simplify'.

Let us suppose that Simon wants to describe this relevant functionality afforded by his software so that other people can use it. He uses our model to describe a *Function F*: to simplify a road network. Simon chooses meaningful names for *F Variables*. The *SoftwareComponent* class that appear on the model ***Figure 2*** is another part of the metadata model that we do not document for java classes. In that case, Simon only writes down an *Activity A* that describes how to carry out *F* with Simplificator class. *A* is illustrated on ***Figure 3***. It is decomposed in two *Activities.* First is *CallJavaMethod* C1 to the method 'characterise' and second is *CallJavaMethod* C2 to the method 'simplify'. In each *CallJavaMethod* objects, Simon specifies the mapping between the *Variables* value and the Simplificator instance fields or method arguments.

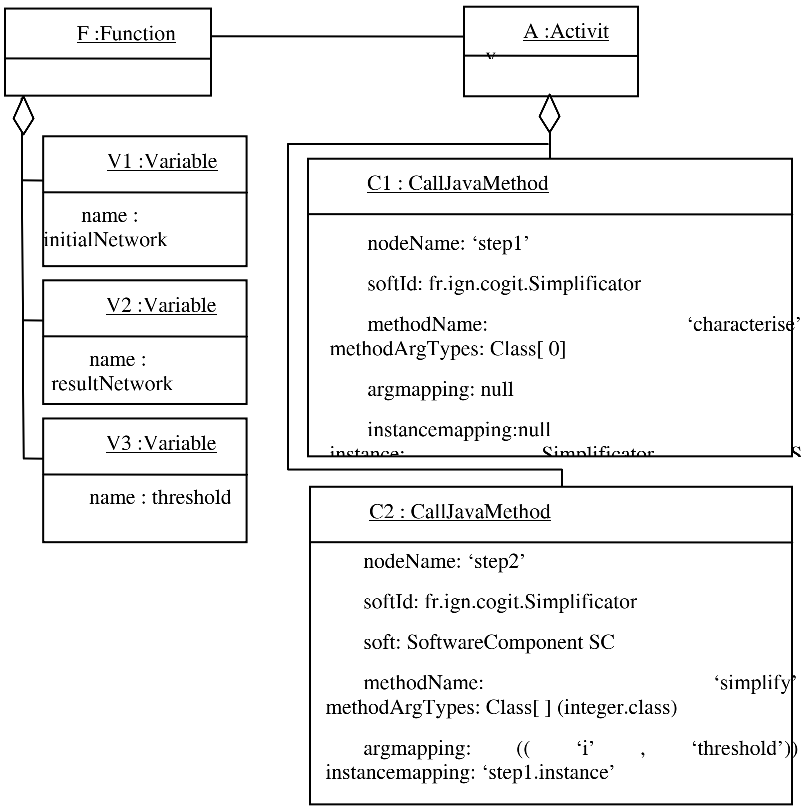

F :Function
A :Activit
y
V1 :Variable
name : initialNetwork
V2 :Variable
name : resultNetwork
V3 :Variable
name : threshold
C1 : CallJavaMethod
nodeName: 'step1'
softId: fr.ign.cogit.Simplificator
methodName: 'characterise'
methodArgTypes: Class[0]
argmapping: null
instancemapping:null
instance: Simplificator S
C2 : CallJavaMethod
nodeName: 'step2'
softId: fr.ign.cogit.Simplificator
soft: SoftwareComponent SC
methodName: 'simplify'
methodArgTypes: Class[ ] (integer.class)
argmapping: (('i' , 'threshold'))
instancemapping: 'step1.instance'

Figure 3. *Detailed structure of the Activity that relates Simon Function to Simon Software, which are detailed on the former figure. The Activity contains two kinds of knowledge: the sequence of java method calls and the mapping between the Function variables and required java objects and fields.*

The authoring application

An initial application to edit metadata about functions provided by existing software has been developed in a PhD work mentioned before (Abd El Kader Y., 2005). This application was a first experiment and another application is being developed based on feedback about this first application. *Figure 4* shows snapshots of this prototype.

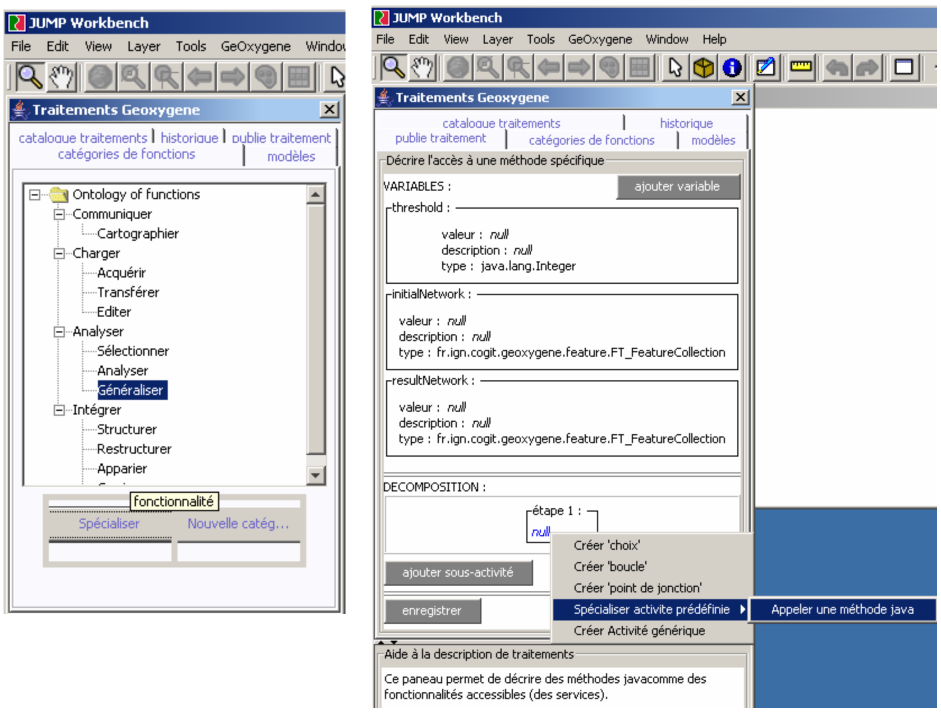
The main criticism was that it was mandatory in this initial application to choose one function category within a predefined list of items. Unfortunately, it was not always easy to identify the most relevant category. That is why in the new interface, some changes have been introduced:

- Author may leave this field blank (even if it is not recommended).
- Author may select more than one category from a list of function categories.

- Initial list of function categories is composed of several classifications that are presented in the following of the section. We keep very generic items to begin with.
- Author may add items to the list of function categories.

It is remarkable that in the field of geospatial information, there is not one consensual classification of GIS functions. There are several types of classifications. None of them covers all aspects covered by the others. Besides, we think it is important to keep the different types of classifications because we expect different people to be keen on different types of classifications. After a review of many existing classifications in the literature, we kept three basic classifications with different organisations. We chose classifications that remained very generic because we fear that too detailed classifications will constrain the authors.

- First classification is project-oriented. It is taken from (Maguire 1991)(Giordano 1994). They identify some basic functions and organise them after four main phases of a typical GIS project: load, integrate, analyse, communicate. Load functions are 'acquire', 'transfer', 'edit'. Integrate functions are 'structure', change structure', match', 'correct'. And so on.
- Second classification is information-oriented. It is taken from (Ormeling 1998) and is adapted to spatial analysis. Ormeling identifies three orders of spatial analysis tasks depending on the manipulated information. First order refers to functions that relate a phenomenon to earth surface. Second order refers to functions that describe relationships between objects belonging to the representation of a same phenomenon (network topology for instance). Third order refers to functions that analyse phenomena.
- Third classification is technique-oriented. It is taken from (Openshaw 1991). It is also adapted to spatial analysis. He concentrates on 'exploratory analysis' and focuses on underlying techniques like fractal analysis, Bayesian mapping, image processing, shape analysis, quadratic methods and so on.



threshold :
valeur : *null*
description : *null*
type : java.lang.Integer

initialNetwork :
valeur : *null*
description : *null*
type : fr.ign.cogit.geoxygene.feature.FT_FeatureCollection

resultNetwork :
valeur : *null*
description : *null*
type : fr.ign.cogit.geoxygene.feature.FT_FeatureCollection

Ce paneau permet de décrire des méthodes javacomme des fonctionnalités accessibles (des services).

Figure 4. Snapshots from our prototype application (French interface). First snapshot shows the simple function categories browser. Second snapshot shows the edition of an activity.

Another important feedback was linked to the description of manipulated data models. Authors did not want to learn a new formalism to express their data model. They wanted the catalogue to interpret their logical schemas automatically. That is why we integrate the PhD work of (Balley, 2006). She proposed a model to describe the structure of a dataset. Main lines of her model are rendered on ***Figure 1*** (see *Structure*). It binds an ontological level to a conceptual level and a logical level. Besides, it parses automatically existing logical schemas (from Geoxygene plate-form only). The physical level is also handled in her model but it is not relevant to our context. The *Structure* model is very interesting in a service-oriented architecture perspective because it supports the description of input and output at several levels of abstraction, which is an important requirement for interoperability. Yet, the author of metadata may choose not to use it and to describe his variables datatypes with simple java class. The graphical user interface of her application must be adapted to our prototype. Based on it, the user will be able to browse existing structures (or ‘models’) and to add a new structure in the list.

CONCLUSION

This paper tackles a specific approach that is not much addressed in existing literature: the acquisition of metadata information about algorithms and processes that will underlie geospatial processing services from the very experts that develop existing software or currently use them.

Important aspects have been highlighted. One is the complexity of the notion of service and of his relation to software and usage. We propose to describe this through a specific metadata model to describe software that encompasses the concept of *Activity*. This model focuses on concepts that are meaningful to experts and are relevant in a service oriented architecture. Another important aspect is the importance of structure description. There exist so far neither models nor tools to describe the structure of data models as it will be needed in service description. This is why we use an ad hoc model developed in a PhD work and the associated editing interface presented in (Balley, 2006).

Another important aspect has not presented in this paper. People are not willing to build and share descriptions of their resources, unless they directly benefit from it or are forced to do so. To meet this issue, we have developed a module that exploits our metadata to simulate service deployment. This module is presented in (Bucher, 2007b). It is Web client that exploits our metadata to display available 'services' and to facilitate their remote invocation and chaining.

Acknowledgement

The authors wish to thank the reviewers for their detailed and precious comments about this paper and work. They also thank Guillaume Touya from COGIT laboratory who helped in improving first drafts of this paper.

BIBLIOGRAPHY

Abd El Kader Y., 2005, Cataloguing Geographical Data Processing Tools, Conception and Exploitation of a Metadata Model, in proceedings of the 22nd International Cartographic Conference ICC 2005, La Coruna, Spain

Albrecht J., 1996, Universal GIS operations for environmental modeling, in proceedings of the 3rd International Conference on Integrating GIS and Environmental Modeling, Santa Barbara, US

Balley S., Bucher B., Libourel T., 2006, A service to customize the structure of a geographical dataset, in proceeding of the Semantic Based GIS workshop, Montpellier, France

Bucher B., Balley S., Richard D., Cébelieu G., Hangouët J.F., 2005 Shareable descriptions of data production processes, in proceedings of the 8th AGILE conference, Estoril, Spain

Bucher B., Balley S., 2007a, A generic preprocessing service for more usable data processing services,.in proceedings of the 10th AGILE Conference, M. Wachowicz and L. Bodum (eds), Aalborg, Denmark

Bucher B., 2007b, A cartographic Web client to explore remote GI methods, in proceedings of the 23rd International Cartographic Conference ICC2007, Moscow, Russia

Craglia M., Kanellopoulos I., Smits P., 2007, Metadata: where we are now, and where we should be going, in proceedings of the 10th AGILE Conference, M. Wachowicz and L. Bodum (eds), Aalborg, Denmark

Giordano A., Veregin H., Borak E., Lanter D., 1994, A conceptual model of GIS-based spatial analysis, in Cartographica, vol 31, n°4

Hubert F., 2002, Map samples to help GI users specify their needs, in proceedings of the 10th International Symposium on Spatial Data Handling, D.E. Richardson et P. van Oosterom (eds.), Ottawa, Canada

Jolivet L., Buard E., Bucher B., Ruas A., 2007, Amélioration de légende sur le Web, actes de la conférence SAGEO, Clermont Ferrand, France

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

Maguire D. J., Dangermond J., 1991, The functionality of GIS, in Geographical Information Systems : Principles and Application, edited by D. J. Maguire, M. F. Goodchild and D. Rhind, (Harlows : Longmans), vol1

Neun, M., Burghardt, D., Weibel, R., 2006, Spatial structures as generalisation support services, in proceedings of the Joint ISPRS Workshop on Multiple Representation and Interoperability of Spatial Data, Hanover, Germany

Open Geospatial Consortium, 2007 a, OGC Web Services Initiative - Phase 5 (OWS-5), Annex B OWS-5 Architecture

Open Geospatial Consortium, 2007 b, OpenGIS® Web Processing Service, OpenGIS® standard

Ormeling F., 1998, Environmental mapping strategies, presented at Intercarto, Brno, Czech Republic

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