

A Rule-based Description Framework for the Composition of Geographic Information Services

Michael Lutz, Roberto Lucchi, Anders Friis-Christensen and Nicole Ostländer

European Commission – Joint Research Centre (JRC)
Institute for Environment and Sustainability, Spatial Data Infrastructures Unit
Via E. Fermi 1, 21020 Ispra, Italy
[michael.lutz | roberto.lucchi | anders.friis | nicole.ostlaender]@jrc.it

Abstract. SDIs offer access to a wealth of distributed data sources through standardised service interfaces. Recently, also geoprocessing capabilities are offered as services in SDIs. Combining data sources and processing services in service chains enable the generation of information that is tailored to the users needs. In this paper, we present a rule-based description framework and an associated discovery and composition method that helps service developers to create such service chains from existing services. The goal of the description framework is to describe services at a conceptual level rather than closely mirroring specific implementation details. It consists of a simple top-level ontology as well as a domain ontology, which provide the basic vocabulary for creating descriptions of both services and the information the service chain is to produce as a result. The composition method uses these descriptions to discover appropriate services and compose them into a service chain that can produce the required information. The method is illustrated using an example from the domain of risk management.

1 Introduction

The main goal of spatial data infrastructures (SDIs) is to offer access to distributed data sources based on the service-oriented architecture (SOA) principle. SDIs provide a framework for optimizing the creation, maintenance and distribution of Geographic Information (GI) services at different organization levels [1] over a distributed computing platform, typically the Web. In such a scenario, where resources are distributed and controlled by different organizations, catalogue services provide a means for describing the services' locations and capabilities. They store meta-information and support users in discovering and using these resources. Consequently SDI-based applications enable an efficient sharing and reuse of geographic data among heterogeneous user groups [1, 2].

Recently, SDIs are also providing capabilities that have traditionally been offered by monolithic GIS [3], including the capture, modelling, manipulation and analysis of geospatial data. In this paper, we are particularly concerned with two kinds of GI services: (1) services that provide geographic data (data access services) and (2) ser-

vices that analyse (and manipulate) geospatial data (geoprocessing services). While standardised interfaces for data access services [4, 5] already exist for several years and have been widely adopted, geoprocessing capabilities have only recently been made available as services in SDIs. A service interface for such services has recently been adopted as a standard by the Open Geospatial Consortium (OGC): The Web Processing Service (WPS) Specification [6].

Since in SDIs data is created by a variety of domain-specific applications, in many cases this data cannot be directly re-used within other domains without further processing. This can be achieved by combining different existing data and geoprocessing services into a value-added *service chain* [7, 8]. Service chains can be described using some orchestration language (e.g. WS-BPEL [9]), which can then be deployed on a corresponding orchestration engine. Creating such service chain descriptions, a task which we term *service composition* in this paper, involves discovering appropriate services for data access and geoprocessing and combining them in a way such that they are capable of generating the required results. Both of these (sub)tasks are currently executed manually by application developers. In this paper we present a method that supports application developers in creating such service chains.

The method is based on a rule-based framework for describing services and the information required by the application developer that stays at the conceptual level and thus abstracts away from specific implementation details. The composition method uses the conceptual service descriptions to discover and (semi)automatically compose a service chain that can provide the information required by the application developer.

In order to illustrate and exemplify the presented methodology throughout the paper, we use the scenario of an application developer, who is requested to deliver a service application that provides information on forest fire density.

The remainder of the paper is structured as follows. In Section 2, we describe previous work in the area of (automatic) service composition and rule-based GI discovery. In Section 3, we present the methodology proposed for supporting the composition of service chains for the given scenario. In Section 4, we conclude the paper and point out topics for future research.

2 Related Work

In this section, we discuss existing approaches for automating the process of service composition and present an approach for discovery of geographic data based on rules, which we extend in this paper to allow also the discovery and composition of services.

Automatic Service Composition. Discovery of services and service functionality is a new task within SDIs. Well-tested specifications of what metadata is required in order to perform this task are lacking. Thus, it can be difficult to understand the functionality of services from their metadata and hence to understand how to combine several services to obtain a certain result [10]. A service composition task can be oriented towards solving different kinds of problems: i) fulfilling preconditions, ii) generating multiple effects and iii) overcoming a lack of knowledge [11]. We are

addressing the latter problem in this paper, i.e. we are concerned with cases where a service providing the required information exists but some of the necessary input parameters are not directly available and, therefore, they must be obtained by using additional services. One possible technique to address this kind of problem is backward chaining [11]. The basic idea of this method is to start by selecting a service which meets the user requirements and place it at the end of the chain (so that it is the last one to be executed). Then, for each input this service requires, services providing such information are added in the chain before the service. The method is iterated until all the necessary input information is available in the chain. In such an approach, what is characterizing the solution is the way user requirements are expressed as well as how the corresponding service selection is done.

In the automatic service composition research line there are a number of related approaches; in the following we mention the ones closely related to the proposal we present. A method for automatic service composition based on backward chaining is presented in [12] where services functionalities are described using ontologies, based on OWL [13] and DAML-S (now OWL-S [14]). Such functionalities are used to express user requirements and, by means of inference engines, to discover and compose services. The semantic descriptions mainly focus on the functionality supplied by the service without expressing in detail the input and output information types as well as the interdependencies between the two. This aspect penalises the approach in scenarios where these details are necessary, for instance in the case where a services will provide different outputs depending on the input provided. E.g., a simple *division* service can return a *density* when its inputs represent a *mass* and a *volume*, or a *velocity* when provided with a *distance* and a *time period*. Furthermore, the types used to describe inputs and outputs are very close to the implementation. Often they are simply modelled as *strings* or simple enumeration types (e.g. *language*). More complex concepts, such as are required to describe spatial data (e.g. feature collections or coverages of a certain type) are not modelled in the examples provided in the application described.

Rule-based GI Discovery. In recent years, ontologies, i.e. formal explicit representations of conceptualizations [15], have been used extensively to model domain-specific knowledge. There are also a number of proposed approaches to use ontologies for the discovery of geographic data [16-19] and GI services in SDIs [10]. Many of these approaches are based on Description Logics (DL) [20] and subsumption reasoning between DL concepts.

In contrast to these approaches, in [21] a *rule-based* approach to discovering data sources within an SDI is presented. It distinguishes two types of rules: *Schema mapping rules* describe a mapping between a local schema and the (global) domain ontology, i.e. they represent the local data using a shared domain vocabulary. *Domain rules* describe domain knowledge, thus complementing the DL concept definitions in the domain ontology. They can then be used to derive implicit knowledge based on existing facts in a knowledge base [22]. Based on these rules, the presented methodology enables the discovery of those data sources within an SDI that contain facts relevant for deriving an answer to the user's question. The rules are traversed backwards, from the goal specified by the user, through domain and schema mapping rules to the data sources. Two alternative approaches for this backward chaining are presented. The

first one only considers class atoms (i.e. atoms representing feature types), while the second one also considers relations.

We base the methodology for service discovery and composition, which is presented in detail in the next section, on the approach presented in [21]. Similarly to this approach, rules are used to describe the resources to be discovered (in this paper, these resources are services rather than data sources). A further similarity is the inclusion of domain rules representing background domain knowledge in the composition approach. Furthermore, both approaches use similar backward chaining approaches, in which the starting point is a goal specified by the user and rules matching (parts of) this goal are used to consecutively discover (and in our case compose) appropriate resources. The approach presented in this paper additionally takes into account how the discovered services have to be combined (while in [21], the order in which data sources are discovered is of no concern). Finally, our approach focuses on (service) descriptions at the conceptual level (while in [21], data sources were described by rules connecting the logical and conceptual levels), and rules are used only for generating a service chain matching the goal specified by the user (rather than also inferring new knowledge as in [21]).

3 An Approach to Semi-Automatic Service Composition

In this chapter, we present a framework for describing services and the information required by the user, together with an associated method for supporting GI service composition.

While existing approaches were strongly focussed on implementation details like the input/output types [12] or very detailed functionality descriptions [10], the goal for the presented description framework and composition method is to stay at the conceptual level. Thus, we want to abstract away from application details at the logical level, which can be very diverse and thus lead to incompatible descriptions if the description follows the logical structure too closely.

In this Section, after giving a general overview (Section 3.1), we present the conceptual description framework (Section 3.2). This consists of a top level ontology of the basic concepts and relations as well as a domain ontology. These are the basic building blocks for creating the description of services and the composition goal. Finally, we describe the composition method based on the presented service and goal descriptions (Section 3.3).

3.1 Overview

In this section, we present an approach that supports service developers in service composition. The goal of the presented approach is to support *generating information* of a particular kind. Hence, we do not consider services that have (real-world) side effects, e.g. reserving a hotel room or charging a credit card, but only services that consume and produce information. In the geospatial domain, these services are either

data access services (e.g. WFS [4], WCS [5]) or geoprocessing services¹ (WPS [6]), which therefore are the focus of our research. For these information generating services, we make the assumption that the output (in relation to the provided input) of such a service can be considered to be the same as its functionality. For example, a service which provides a (Euclidian) distance between two points has the functionality “compute Euclidian distance”. In contrast, a hotel booking service could have the functionality of booking a hotel room (and charging your credit card) while returning a booking confirmation document as an output. We therefore propose a method for supporting service composition that focuses on *service outputs*.

We adopt a backward chaining approach (cf. Section 2) for supporting the user in composing a service chain. In the following, we give a more detailed overview of this approach (Fig. 1).

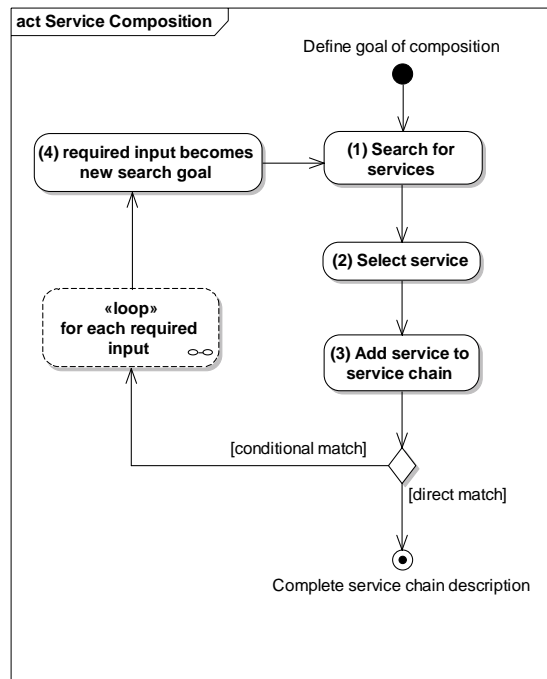


Fig. 1: UML activity diagram illustrating the steps of the proposed approach

The basis for the method is a conceptual description framework. The framework will be used to describe both the services available in the infrastructure and the information required by the user, i.e. the output of the application to be created. The latter is goal and starting point for the service composition method. The description framework is presented in Section 3.2.

¹ While the WPS specification allows arbitrary (i.e. also non-spatial) processing to be provided by a WPS, we restrict our focus to those WPS that provide spatial data as output.

The first step in the composition process is to search for services that can generate information as specified by the user. In some cases, data access services providing such data might be directly available (*direct match* in Fig. 1). In most cases, however, only geoprocessing services will be available that can *potentially* generate the required data, *provided that they are given appropriate input data* (*conditional match* in Fig. 1). In this case, further searches have to be performed to discover services that can provide appropriate input data.

In each discovery step, the user has to select one of the discovered services (step 2 in Fig. 1). In order to help the user in selecting the most appropriate service, the discovered services could be ranked. The ranking could be based on specific service properties that are specified by the user during the request formulation, e.g. service security or trust in a certain service or service provider. In addition, some heuristics for the quality of the match can be used, e.g. some measure of how similar the output of a discovered is to the goal. Developing such methods for ranking the discovered services will be part of our future research.

The selected service is then added to the service chain description (step 3). The service is connected to the service already present in the chain for whose input the search was performed. Thus, the service chain is built back-to-front.

If the discovered service requires additional input data, these become the search goal for these further searches (step 4). The discovery (steps 1-3) has to be repeated, until data access services have been discovered for all required input data.

3.2 A Conceptual Description Framework for Services and Composition Goal

Discovery in the presented composition approach is based on a conceptual description framework for both the composition goal and the services available in the infrastructure. For this description framework, we extend the rule-based approach for GI discovery presented in Section 2. In the following sections, we define a simple top level ontology for service descriptions and illustrate how a domain ontology can be derived for a certain domain of interest. Finally, we show how concepts and relations from these ontologies can be used to create service descriptions.

In this chapter and the remainder of the paper, we use the following basic terminology. A *term* is a constant or variable symbol. In this papers, constants are written starting in upper case (e.g. *Event*), while variables start in lowercase (e.g. *ec*). If R is a predicate symbol of arity n and $t_1; \dots; t_n$ are terms, $R(t_1; \dots; t_n)$ and $\neg R(t_1; \dots; t_n)$ are called *literals*. A *rule* is a disjunction of literals with *exactly* one positive literal and at least one negative literal, i.e. it has the form $h \vee \neg b_1 \vee \dots \vee \neg b_m$ which is equivalent to $b_1 \wedge \dots \wedge b_m \rightarrow h$. $b_1 \wedge \dots \wedge b_m$ is called the *body* of the rule, and h is called the *head*. Note that in this paper, we also use rules with more than one head literal. As these head literals do not include existential quantification, a rule of the form $b_1 \wedge \dots \wedge b_m \rightarrow h_1 \wedge \dots \wedge h_n$ can be trivially reduced, without loss of meaning, to n rules of the form: $b_1 \wedge \dots \wedge b_m \rightarrow h_1$; $b_1 \wedge \dots \wedge b_m \rightarrow h_2$; \dots ; $b_1 \wedge \dots \wedge b_m \rightarrow h_n$. A rule with one positive literal and no negative literals, i.e. one that only contains a head and no body, is called a *fact*. [23]

Top level ontology. For the limited domain considered for our method (only data access and geoprocessing services), we have developed a simple top level ontology, which provides a basis for the domain ontology (described in the next section) as well as the service description rules and the specification of a composition goal by the user. The main concepts in this ontology mirror the duality between the field and object views of geographic information [3]. Thus, we distinguish the following main concepts (unary predicates) and relationships (n-ary predicates):

- **entity.** Unary predicate denoting an entity (or object)
- **entityCollection.** Unary predicate denoting a collection of entities.
- **field.** Binary predicate denoting a field (or spatio-temporal function). The first argument is the (value) range of the field; the second argument is its (spatio-temporal) domain.
- **hasMember.** Binary predicate stating that an entity collection contains an entity.

The top level ontology is deliberately kept simple. It will be extended and aligned with geographic top level ontologies such as presented in [24] and [25] as part of our future work.

Domain ontology. Further rules describing relationships in a domain of interest can be considered in the methodology (called domain rules in [21]). These describe types of entities and fields used in the domain and the relationships between them. They can also include implicit domain knowledge that can be used for inferences during discovery and composition. Domain rules can thus be used as a shared vocabulary [26] by users and service providers. Note that while currently domain rules are not labelled, labels could be used to mark different sets of domain rules (e.g. from different user communities). These labels could then be used to identify the sets of rules used for deriving the service chain.

In Fig. 2, some examples for domain rules are given. Lines 1 and 2 state that a *density* d (respective a *frequency* f) of an entity collection ec with respect to a spatio-temporal domain std is a *field* with f as its range and std as its domain. Line 3 asserts that a *forestFire* is an *entity*. Lines 4 and 5 state that entity collections whose members are *adminUnits* (respective *postCodes*) can be considered as *tessellations*.

- 1 $\text{density}(d, \text{std}, ec) \rightarrow \text{field}(d, \text{std})$
- 2 $\text{frequency}(f, \text{std}, ec) \rightarrow \text{field}(f, \text{std})$
- 3 $\text{forestFire}(e) \rightarrow \text{entity}(e)$
- 4 $\text{entityCollection}(ec), (\forall e : \text{hasMember}(ec, e) \rightarrow \text{adminUnit}(e)) \rightarrow \text{tessellation}(ec)$
- 5 $\text{entityCollection}(ec), (\forall e : \text{hasMember}(ec, e) \rightarrow \text{postCode}(e)) \rightarrow \text{tessellation}(ec)$

Fig. 2: Examples for domain rules: *density* and *frequency* are *fields* (1-2); *forestFires* are *entities* (3); all feature collections that contain *adminUnit* or *postCode* features are *tessellations* (4-5).

Service descriptions. In the approach presented in [21], the focus was on describing feature types provided by data access services and general domain knowledge. In contrast, in this paper, the goal is to describe operations provided by data access or

geoprocessing services. An operation is described by a rule specifying the relationship between the operation's inputs and its outputs. Such a rule can be interpreted as follows: "If input of a certain type is available, then (using the operation associated with this rule) output of a certain type can be provided". To enable the association of a rule with an operation, the rules are labelled. These labels will later also be used to link the operation description to (other) service metadata stored in a catalogue service (cf. Section 4).

One basic assumption for our approach is that at the logical level we only consider services that provide features, feature collections or coverages as outputs (cf. Section 3.1). The services naturally also need to be able to consume these types of parameters, as well as simple datatypes like strings or integers. In this paper, we consider operation signatures at the logical level only for illustration purposes. We have therefore chosen a simplified representation that is not based on a formal model like the top level ontology used for the descriptions at the conceptual level. Such a formal model allowing to refer to the actual XML datatypes used in the implementations (e.g. GML types like `gml:AbstractFeatureCollectionType` or simple XML schema datatypes like `xs:string`) will be developed as part of our future work. This model will then be used for expressing mappings from the logical to the conceptual level.

Some examples for operation signatures that are useful in the context of our example scenario are given in Fig. 3. The *getFeature* operation (of a WFS) returns a feature collection of a certain feature type for a given query. Similarly, the *getCoverage* operation (of a WCS) returns the requested coverage based on the given query. The other three operations shown in Fig. 3 are geoprocessing operations. The *count* operation counts for each feature *f* in *fc2* the number of features of *fc1* that are spatially contained within *f*. The count value is added as an attribute to *f* and the new feature collection is returned as a result. The *normalizeByArea* operation normalizes (for each feature *f* in the feature collection *fc*²) the value of the attribute identified by the parameter *attToNorm* by the area of the geometry of *f*. The normalized value is added as an additional attribute to the feature collection, which is returned as a result.

- 1 `getFeature(featureType:string, q:query):FeatureCollection`
- 2 `getCoverage(coverage:string, q:query):Coverage`
- 3 `count(fc1:FeatureCollection, fc2:FeatureCollection):FeatureCollection`
- 4 `normalizeByArea(fc:FeatureCollection, attToNorm:string):FeatureCollection`

Fig. 3: Examples for operations at the logical level.

Example descriptions for each of these operations at the conceptual level are shown in Fig. 4 and Fig. 5. Note that, based on the assumption described above, only relationships between those parameters of an operation that can be considered as entities, entity collections or fields are described at the conceptual level. The inputs and outputs are described using concepts from a domain ontology, e.g. *density* or *frequency*.

Data access services and geoprocessing services are described slightly differently. While every instance of a processing service will exhibit the same functionality, i.e.

² *fc* is assumed to consist of polygon features forming a tessellation.

provide the same output given the same input, different instances of a data access service will provide different outputs, depending on the feature types or coverages stored in its underlying data store. Therefore, data access services have to be described at the instance level, while geoprocessing services can be described at the type level.

For data access services, different rules describe the different feature types and coverages that can be provided by a specific instance of a WFS or WCS. As these services do not have any inputs (that are included in the description at the conceptual level) they can be described as facts, i.e. as rules without a body. For example, the rules in Fig. 4, lines 1-2 state that the described WFS *getFeature* operation can provide *entityCollection* all of whose members are *forestFires* and *adminUnits*. The rule in line 3 describes a WCS whose *getCoverage* operation can provide an *elevation* field (with domain *d* and range *e*). Feature types and field types should already be defined in the domain ontology as rules such as those shown in Fig. 2, lines 1-3.

- 1 **getFeature:** $\rightarrow \text{entityCollection}(\text{ec}), (\forall e : \text{hasMember}(\text{ec}, e) \rightarrow \text{forestFire}(e))$
- 2 **getFeature:** $\rightarrow \text{entityCollection}(\text{ec}), (\forall e : \text{hasMember}(\text{ec}, e) \rightarrow \text{adminUnit}(e))$
- 3 **getCoverage:** $\rightarrow \text{elevation}(e, d)$

Fig. 4: Examples for service description rules for data access services

Also, for geoprocessing services, there can be several rules describing the same operation. Each of these rules expresses a different conceptualization. Thus, two rules could e.g. express that the *normalizeByArea* operation can be used to derive a *density* from a *frequency* (line 2 in Fig. 5) as well as a *fraction* from an *area* (line 3).

- 1 **count:** $\text{entityCollection}(\text{ec1}), \text{tessellation}(\text{ec2}) \rightarrow \text{frequency}(f, \text{ec2}, \text{ec1})$
- 2 **normalizeByArea:** $\text{frequency}(f, \text{std}, \text{ec}) \rightarrow \text{density}(d, \text{std}, \text{ec})$
- 3 **normalizeByArea:** $\text{area}(a, \text{std}) \rightarrow \text{fraction}(f, \text{std})$

Fig. 5: Examples for service description rules for geoprocessing services

The most intuitive mapping of types at the logical to types at the conceptual level would be from feature collections to entity collections and from coverages to fields. However, there might also be cases where a parameter, while represented as a feature collection at the logical level, can also be conceptualized as a field, or, more rarely, where a coverage can be represented conceptually as an entity collection. We will illustrate this using the *normalizeByArea* operation listed in Fig. 3, line 4. Conceptually, the input of the operation could also be seen as a coverage, whose domain consists of the set of geometries of all features in *fc*, and whose range is the set of the corresponding attribute values of *attToNorm*.

We believe that feature collections are best mapped to *fields* in this way if only one attribute of its features is of interest for the processing (in the case of the collection being an input parameter) or for the result of the processing (if the collection is the output). Conversely, a feature collection should be mapped to a (subconcepts of) *entityCollection* if its features are used as a whole in the processing, e.g. the features in *fc1* in the *count* operation (Fig. 3, line 3). If new features (entities with a different

identity) are created in the processing, e.g. a buffer zone around a feature, this output feature collection should also be mapped to an *entityCollection*.

The representation of an operation's parameters as a field or entity collection will of course affect its discovery. To alleviate this decision, rules could be included that map between both views (e.g. "if *ec* is an *entityCollection*, and every entity *e* that is a member of *ec* has a *temp* property *t* and a *geometry* property *g*, then the collection of all pairs (*t,g*) is a *temp* field"). Such statements will also need to be part of the mapping rules between the logical and conceptual level that we will develop as part of our future work.

3.3 Rule-based Service Discovery and Composition

Based on the description framework described in the previous section, services can be discovered and composed into a service chain that is able to produce the information defined as the composition goal by the user. To illustrate the presented method, we show how a service chain can be composed for the scenario introduced in Section 1. For this walk-through, we assume the service description rules listed as examples in Fig. 4 and Fig. 5 and the domain rules given in Fig. 2.

Goal description. It is the aim of the proposed composition method to create a service chain that can provide the information required by the user without requiring any further input. This means that the chain behaves like a data access service, and therefore the composition goal is specified in the same way, i.e. as a fact. The application developer's goal ("forest fire density") can be represented as a density *d* of an entity collection *ec* (all of whose members are *forestFires*) with respect to a spatio-temporal domain *std*:

$$\text{density}(d, \text{std}, \text{ec}), (\forall e : \text{hasMember}(\text{ec}, e) \rightarrow \text{forestFire}(e))$$

Discovery and composition. During each discovery step, the current goal is compared with the rules describing the services. A rule is considered to be a match for a goal if it contains (part of) the goal in its head. Rules that have no body (i.e. describe data access services) represent direct matches and are endpoints in the composition process (cf. Section 3.1). Rules that have a body represent conditional matches and require additional searches. The head literal is replaced by the body literals, and the thus generated rule becomes the new goal. When there is a match (and the rule is not a domain rule), the corresponding operation (which can be derived from the label) is added to the service chain.

Fig. 6 shows each of the discovery steps, in which both (labeled) service description rules and a domain rule are used.

The first query for this goal to the does not return any direct matches. However, a conditional match would be discovered: The *normalizeByArea* operation – under the condition that forest fire *frequency* data can be provided as input. This operation is added to the service chain, and the new goal for the next query is defined as a fre-

quency f of an entity collection ec (all of whose members are *forestFires*) with respect to a spatio-temporal domain std :

$$\text{frequency}(f, \text{std}, ec), (\forall e : \text{hasMember}(ec, e) \rightarrow \text{forestFire}(e))$$

This search only returns one (conditional) match, the *count* operation – under the condition that a collection of *forest fires* and a feature collection that represents a *tessellation* of space can be provided as inputs:

$$\text{tessellation}(\text{std}), \text{entityCollection}(ec), (\forall e : \text{hasMember}(ec, e) \rightarrow \text{forestFire}(e))$$

When searching for these two data sets, two direct matches are discovered: The *getFeature* operations providing *forestFire* and *adminUnit* entity collections. Note that for discovering the *adminUnit* entityCollection, a domain rule (stating that *adminUnit* entity collections represent *tessellations*) has to be used as an intermediate step.

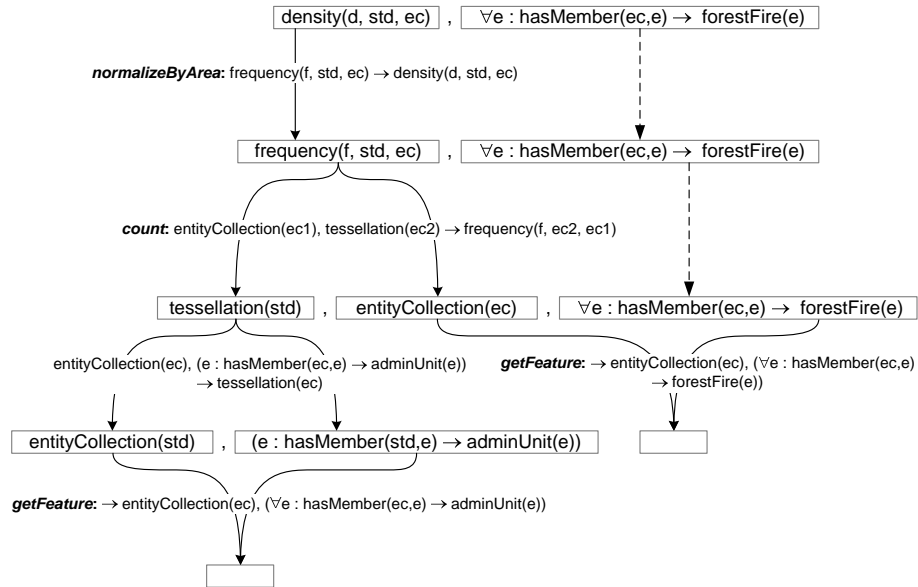


Fig. 6: Example composition of a service chain providing forest fire density information.

The data flow in the resulting service chain is depicted in Fig. 7. Note that as our composition methodology only uses the service descriptions at the conceptual level, it does not create a service chain description that is directly executable using some workflow engine. Deriving such an executable description, which might include further (e.g. coordinate or schema) transformation steps between each of the component services, will be part of future work.

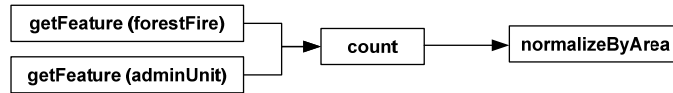


Fig. 7: Data flow in the composed service chain.

4 Conclusion & Future Work

In this paper, we have presented a method for supporting users in creating service chain descriptions on a conceptual level. The method builds on a rule-based description framework for data access and geoprocessing services. We have illustrated how this framework can be used to describe service functionalities on a conceptual level by linking their inputs and outputs. The presented method uses these descriptions to semi-automatically derive a service chain description. The presented approach will be extended in several directions in our future research:

- **Creating executable service chain descriptions.** In this paper, we only consider composition at the conceptual level, at which services are described. This service chain description cannot directly be executed by a workflow engine. To achieve this, a formal model for mapping the operations at the logical level to the descriptions at the conceptual level has to be developed. An important requirement for the mapping language to be developed is to (intuitively) support mapping between the field and entity views on GI. The presented composition method will be extended to take these mappings into account in order to create an executable service chain description.
- **SDI architecture.** Based on this extended composition method, an SDI architecture and prototype will be developed. Most importantly, these will combine the service discovery steps at the conceptual level with service discovery using catalogue services. To realize this, a suitable reference between conventional metadata and the rule-based descriptions. E.g., when using ISO 19119 metadata [8], this reference could be included in the *operationDescription* attribute of the *SV_OperationMetadata* item [27].
- **Enhancing ontologies.** In this paper, we have presented a simple top level ontology as a basis for the description framework. This ontology should be compared and, if possible, aligned with other geographic top level ontologies (e.g. [24, 25]). Also, we have not yet developed a full-fledged domain ontology, which is the main basis for service providers to describe their services' functionalities (and therefore a crucial factor during discovery). The presented concepts will be extended with further examples from the domain of disaster management (e.g. flood damage assessment).
- **Enhancing the discovery and composition method.** Finally, the presented composition method can be enhanced in several ways. In order to avoid introducing loops in the service chain, services that are already part of the service

chain (in the same branch) should not be presented as results during discovery³. Also, heuristics could be introduced for ranking the discovered services and thus helping the user in selecting the most appropriate service. For example, several further discovery steps could be automatically executed (in the background) to find out how many more services would have to be added to the discovered service before a direct match is found. Other ranking criteria could be some measure of how similar the output of a discovered is to the goal. For such a ranking, a similarity metric would have to be developed.

Acknowledgements

The work presented in this paper has been supported by the European Commission through the ORCHESTRA project (grant number IST-2002-511678).

References

1. Nebert, D. D.: Developing Spatial Data Infrastructures: The SDI Cookbook. GSDI (2004)
2. McKee, L.: Who wants a GDI? In: R. Groot and J. McLaughlin, (eds.): Geospatial Data Infrastructure - Concepts, cases, and good practice. Oxford University Press (2000) 13-24
3. Worboys, M., Duckham, M.: GIS – A Computing Perspective. CRC Press (2004)
4. OGC: Web Feature Service (WFS) Implementation Specification, Version 1.1. Open Geospatial Consortium, OGC 04-094 (2004)
5. OGC: Web Coverage Service (WCS) Implementation Specification, Version 1.1. Open Geospatial Consortium, OGC 06-083r8 (2006)
6. OGC: Web Processing Service (WPS) Implementation Specification, Version 1.0.0. Open Geospatial Consortium, OGC 05-007r6 (2007)
7. ISO: Information Technology - Open Distributed Processing - Reference Model: Overview. International Organization for Standardization, ISO/IEC 10746-1 (1998)
8. ISO: Geographic Information - Services. International Organization for Standardization, ISO 19119:2005 (2005)
9. OASIS: Web Services Business Process Execution Language (WS-BPEL), Version 2.0. Organization for the Advancement of Structured Information Standards (2007)
10. Lutz, M.: Ontology-based Descriptions for Semantic Discovery and Composition of Geoprocessing Services. *GeoInformatica* 11 (2007) 1-36
11. Küster, U., Stern, M., König-Ries, B.: A Classification of Issues and Approaches in Automatic Service Composition. In: Proc. First International Workshop on Engineering Service Compositions (WESC05) at Third International Conference on Service Oriented Computing (ICSOC05) (2005)
12. Sirin, E., Hendler, J., Parsia, B.: Semi-automatic Composition of Web Services using Semantic Descriptions. In: Proc. 1st Workshop on Web Services: Modeling, Architecture and Infrastructure (2002)
13. Antoniou, G., Van Harmelen, F.: Web Ontology Language: OWL. In: S. Staab and R. Studer, (eds.): Handbook on Ontologies. Springer (2003) 67-92

³ The same service occurring in parallel branches is not a problem.

14. Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., Parsia, B., Payne, T., Sabou, M., Solanki, M., Srinivasan, N., Sycara, K.: Bringing Semantics to Web Services: The OWL-S Approach. In: Proc. First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004) (2004)
15. Gruber, T.: A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition* 5 (1993) 199-220
16. Klien, E., Lutz, M., Kuhn, W.: Ontology-Based Discovery of Geographic Information Services - An Application in Disaster Management. *Computers, Environment and Urban Systems* 30 (2006) 102-123
17. Lutz, M., Klien, E.: Ontology-Based Retrieval of Geographic Information. *International Journal of Geographical Information Science* 20 (2006) 233-260
18. Bowers, S., Ludäscher, B.: An Ontology-Driven Framework for Data Transformation in Scientific Workflows. In: Proc. International Workshop on Data Integration in the Life Sciences (DILS'04) (2004)
19. Yue, P., Di, L., Yang, W., Yu, G., Zhao, P.: Semantics-based Automatic Composition of Geospatial Web Service Chains. *Computers & Geosciences* 33 (2007) 649-665
20. Baader, F., Nutt, W.: Basic Description Logics. In: F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, (eds.): *The Description Logic Handbook. Theory, Implementation and Applications*. Cambridge University Press (2003) 43-95
21. Lutz, M., Kolas, D.: Rule-based Discovery in Spatial Data Infrastructures. *Transactions in GIS, Special Issue on the Geospatial Semantic Web* 11 (2007) 317-333
22. Kolas, D., Hebler, J., Dean, M.: Geospatial Semantic Web: Architecture of Ontologies. In: Proc. First International Conference on GeoSpatial Semantics (GeoS 2005) (2005)
23. Russell, S. J., Norvig, P.: *Artificial Intelligence. A Modern Approach*. Prentice-Hall, Upper Saddle River, NJ (USA) (2003)
24. Bittner, T.: From Top-level to Domain Ontologies: Ecosystem Classifications as a Case Study. In: Proc. Conference on Spatial Information Theory (COSIT 2007) (2007)
25. Probst, F., Espeter, M.: Spatial Dimensionality as Classification Criterion for Qualities. In: Proc. International Conference on Formal Ontology in Information Systems (FOIS 2006) (2006)
26. Wache, H., Vögele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., Hübner, S.: Ontology-Based Integration of Information — A Survey of Existing Approaches. In: Proc. IJCAI-01 Workshop: Ontologies and Information Sharing (2001)
27. Lutz, M.: Ontology-Based Service Discovery in Spatial Data Infrastructures. In: Proc. ACM Workshop on Geographic Information Retrieval (GIR'05) (2005)