Replication of the Question-Based Spatial Computing Approach – Experiences and Suggestions for Further Developments

GI_Forum 2019, Issue 1
Page: 40 - 53
Full Paper
Corresponding Author:
studer.selina@gmx.ch
DOI: 10.1553/giscience2019_01_s40

Selina Studer and Barbara Hofer
University of Salzburg, Salzburg, Austria

Abstract

Geographic Information Systems (GIS) have developed into complex toolboxes and require analysts to formulate spatial questions according to the requirements of the data formats and tools provided by their specific GIS-application. The recently proposed language for spatial computing aims to provide a question-based and thus more comprehensible approach for spatial analyses that especially supports scientists and experts from other disciplines to conduct spatial analyses in their fields. In this contribution, we apply the question-based spatial computing approach to a case study in the humanitarian field and compare the resulting script to one written using a conventional GIS tool. The comparison of the two versions of the script is based on six criteria covering qualitative and quantitative aspects of the analysis. We also discuss the implementation concept behind the new language. Our results show that the new approach requires fewer computational steps than the conventional script. In addition, the declarative approach allows users to focus on the content of the spatial question, and the query-like character of the language makes it easier to understand for non-GIS specialists. In addition, we share observations on challenges of the further development of the language as an outcome of this study.

Keywords:

core concepts; language for spatial computing; question-based analysis; domain-specific language; transdisciplinarity

1 Introduction

Spatial analyses are motivated by the need to find an answer to a question; they thus help to understand problems better and to support decision making. This is why spatial analyses play an important role in various disciplines, including the humanitarian field. The tools for spatial analyses, Geographic Information Systems (GIS), have been developed into complex toolboxes, requiring analysts to formulate spatial questions according to the requirements of the data formats and tools provided by their particular GIS-applications. Deciding which tools and data to use to answer spatial questions distracts users from the core of the question, requires expert knowledge (Albrecht, 1989; Kuhn, 2012; Scheider, Ballatore & Lemmens, 2018), and limits (intellectual) resources and time for critical spatial thinking

(Bearman, Jones, André, Cachinho & DeMers, 2016). This prevents non-GIS specialists from carrying out spatial analyses effectively, and consequently reduces the use of spatial analyses for knowledge generation in potential GIS application domains (Vahedi, Kuhn & Ballatore, 2016).

Counteracting this situation, Kuhn and Ballatore (2015) developed a question-based approach for spatial analyses that uses the *language for spatial computing* (Kuhn & Ballatore, 2015). This language builds on seven core concepts for spatial information, through which space is perceived in a content-related way. The query-like nature and content orientation of the language simplify spatial analyses (see Section 2). The idea behind the language was illustrated in Vahedi et al. (2016). They demonstrated how the structured query language (SQL) allows the querying of relational databases in the form of simple questions by using the semantics 'SELECT *attributes* FROM *tables* WHERE *condition*'. The same aim exists for spatial questions, which must be answered with simple and normative semantics in a content-oriented way. Like SQL, the language for spatial computing should be used across disciplines; it should achieve a mature self-image and understanding of its tools (Vahedi et al., 2016).

The language for spatial computing has been implemented in some initial case studies, and eventually it should result in a high-level programming language that can be used on existing GIS platforms (Vahedi et al., 2016). Kuhn and Ballatore (2015) called for further research to clarify whether the underlying core concepts are sufficient to form the language for spatial computing, and to examine how core computations of the language for spatial computing mediate between underlying GIS applications and the core concepts. Based on the resulting findings, formal specifications of the language for spatial computing and software integration could be revised (Kuhn & Ballatore, 2015).

In this contribution, we apply the language for spatial computing to a real-world case in the humanitarian field and compare the resulting Python script to a script containing the conventional ArcPy-analysis. The approach followed is compared to the work presented in Vahedi et al. (2016) and contributes to the request for further research made by Kuhn and Ballatore (2015). We also use Vahedi et al.'s (2016) Python implementation of the language for spatial computing and implement additional computations for our case study. In order to determine the criteria for comparison in our research, we extend the criteria used in Vahedi et al. (2016) and use the following six criteria to assess the simplification achieved through the language for spatial computing: *question-based*, *computational steps*, *comprehensibility*, *role of base language*, *role of underlying GIS*, *role of data property*.

The replication of a case study using the question-based spatial computing approach allows us to investigate two questions:

(1) Can similar conclusions about the benefits of the language be reached for the present application case in comparison to what was reported in Vahedi et al. (2016)?
(2) What experience is gained through the use of the language for spatial computing that allows suggestions to be made for its further development?

2 Core concepts for spatial information and the language for spatial computing

Janelle and Goodchild (2011) identified the need for a clear and simple conceptual view to understand geoinformation. Kuhn (2012) built on this work, which captured spatial phenomena in a few abstract concepts, and suggested that space is perceived through seven core concepts of spatial information (Table 1). Unlike the geo-atom of Goodchild, Yuan and Cova (2007) that abstracts spatial phenomena to a single form, the core concepts are a content-based abstraction of spatial information. The abstraction level was chosen to be as high as possible in order for one to be able to grasp all the concepts at once while still allowing one to make sense (Kuhn, 2012). The concepts of granularity and accuracy (nos. 6–7 in Table 1) can be applied as quality concepts to the content concepts of location, field, object, network and event (nos. 1–5).

Table 1: Overview of the core concepts for spatial information (Kuhn & Ballatore, 2015).

No.	Core Concept	Question	
1	Location	where	*Content concepts*
2	Field	value of a position in space and time	
3	Object	its properties and relations to other objects	
4	Network	connectivity between objects	
5	Event	time or duration in fields, object or network	
6	Granularity	amount of detail in fields, objects, network and events	*Quality concepts*
7	Accuracy	accuracy of information with respect to a reference	

The core concepts of spatial information are the underlying concept of the language for spatial computing (Vahedi et al., 2016). The language is structured in such a way that geodata is read as one of the core concepts, depending on the content and the question to be answered. This turns imported geodata into Abstract Data Types (ADTs[1]); data can be seen as instances of the core concepts available for manipulation using the language for spatial computing. For each core concept, a set of *core computations* exist that correspond to analysis functionality (Vahedi et al., 2016). The linking of the data with core concepts suggests which core computations can be applied to the data.

A set of only a few meaningful and combinable core computations reduces the complexity of spatial analyses and allows users to speak a language they are familiar with, unlike the technical languages that software often requires (Kuhn & Ballatore, 2015). Thereby, spatial analyses shift from spatial computing to answering questions, which enhances the transdisciplinary use of spatial analyses (Hofer & Scheider, under review; Kuhn, 2012).

[1] ADT: 'class of object whose logical behaviour is defined by a set of values and a set of operations' (Dale & Walker (1996), in Vahedi et al. (2016)).

The core computations are a layer implemented on top of existing GIS applications such as ArcGIS. Thus, the language for spatial computing mediates between the technological layer and the user's perception of spatial information (see Figure 1). The core computations have been implemented in different languages, including Python (Kuhn & Ballatore, 2015).

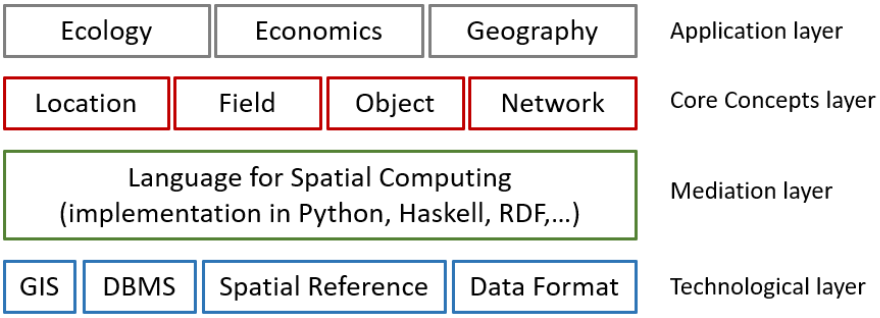

Figure 1: The core concepts and their implementation using the language for spatial computing mediate between the technological layer and the user (application layer) (adapted from Kuhn and Ballatore (2015)).

Vahedi et al. (2016) applied the language for spatial computing to a real example from economics. They used Python to implement the language in the form of a Python library named *CoreConcepts*. The library consists of a Python class for each core concept, and the core computations were implemented as Python methods of these classes using the ArcPy library. The authors compared a conventional analysis using ArcPy with an analysis using the language for spatial computing. They highlighted the declarative approach of the new language versus the procedural solution of the conventional analysis, and showed a 45% reduction in the computational steps achieved using the language for spatial computing.

3 Case study and evaluation criteria

This work investigates whether the findings of Vahedi et al. (2016) – the reduction in the number of computational steps, and the simplified understanding of spatial analysis using the language for spatial computing – can be confirmed by another case study. This section introduces a real-world case study, and the selection of criteria for the assessment of the analysis implemented using the language for spatial computing.

3.1 The case study

The case study is taken from the humanitarian field, where crisis managers often work in interdisciplinary teams and make judgements and decisions under pressure (Cai, Sharma, MacEachren & Brewer, 2006). The humanitarian sector is thus a prime example of a field that has a need for a simple, established language for spatial computing. The spatial analysis is borrowed from the International Committee of the Red Cross (ICRC): an examination of the access of households to water sources that has been used by engineers in the field (ICRC,

2017). The purpose of our analysis was to identify buildings adequately supplied by water points according to distance and elevation parameters. The input data consisted of a Digital Elevation Model (DEM)[2], feature datasets with water points[3], buildings[4], the area of interest, and the two parameters distance and elevation. The results are a feature layer for each water point showing the buildings which lie within the distance and elevation parameters (visualized in Figure 2). Access to water is an important factor for behavioural responses to hygiene and sanitation measures (Ntozini et al., 2015), and it is therefore frequently analysed in the humanitarian field. The case study is hereafter referred to as the 'urban water analysis'.

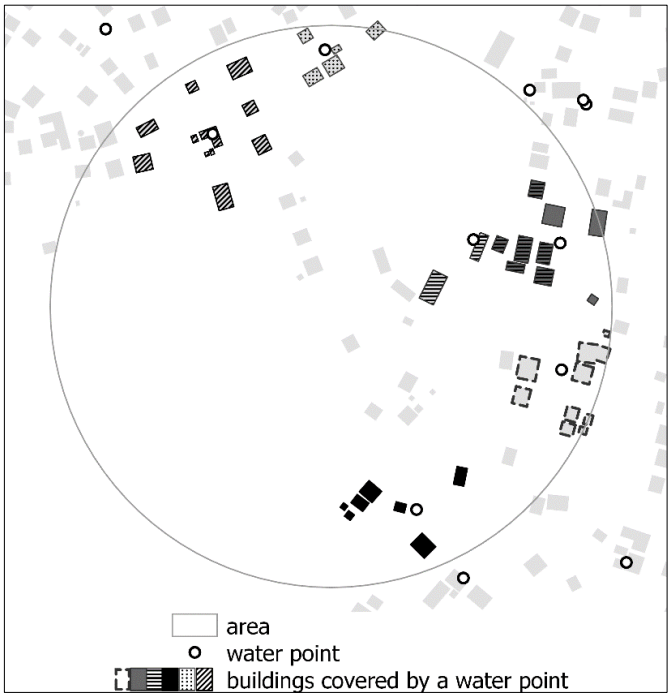

Figure 2: Input data and the resulting buildings falling within desired distance and elevation criteria.

The urban water analysis was conducted twice: once using ArcPy (conventional analysis), and once using the language for spatial computing. The Python scripts of the conventional analysis and of the implementation based on core concepts can be found in Figure 3. The procedural steps[5] of the conventional urban water analysis (see left side of Figure 3) are:

[2] SRTM, retrieved from http://dwtkns.com/srtm30m/, accessed 7.12.2018

[3] Akvo National water point mapping Sierra Leone, Retrieved from Humanitarian Data Exchange (HDX): https://data.humdata.org/dataset/national-water-point-mapping-sierra-leone, accessed 7.12.2018

[4] OpenStreetMap Building export, retrieved from HDX: https://data.humdata.org/dataset/hotosm_sierra_leone_buildings, accessed 7.12.2018

[5] The original analysis from ICRC was adapted and shortened for greater clarity.

(1) Select water points within area,
(2) select buildings within area,
(3) extract elevation values of water points to the water points attribute table,
(4) calculate centroids of buildings,
(5) extract elevation values of the buildings' centroids to the buildings attribute table,
(6) join the attributes with the elevations to a copy of the building feature class,
(7) select buildings within the distance and elevation parameters with a for-loop through each water point.

For the implementation using the language for spatial computing, the well-documented Python implementation was used (Kuhn & Ballatore, 2015; Vahedi et al., 2016, and on GitHub[6]). The backend implementation of the core computations was implemented using ArcPy. Code developed within this research is also available on GitHub[7] and therefore supports reproducible research as defined by Nüst et al. (2018).

3.2 Criteria for assessing the simplification of the spatial analysis

To assess the benefits and constraints of the language for spatial computing, the question-based analysis was assessed and compared with the conventional analysis. The comparison suggested in Vahedi et al. (2016) considered the criteria *question-based*, *computational steps* and *role of data property*. In this contribution, we extend these criteria in order to consider further aspects of the language in the evaluation. The extension of the criteria takes every layer of the conception of the language into consideration (see Figure 1). Table 2 shows the six criteria and how they are linked to the four layers.

Table 2: Overview of the criteria used for the comparison of the analyses.

Layer	Criteria
Application layer	Question-based
	computational steps
Core concept layer	Comprehensibility
Mediation layer	Role of base language
Technological layer	Role of underlying GIS
	Role of data property

The criteria *role of underlying GIS*, *role of data property* and *role of base language* examine the effect of the initial technical set-up (consisting of ArcGIS and Python) on the implementation of the language for spatial computing. The criterion *computational steps* considers the difference in the number of computations between the conventional analysis and the analysis that uses the core concepts. Using the qualitative criteria *question-based* and *comprehensibility*, we assess in what sense the language for spatial computing simplifies spatial analyses, for which we

[6] https://github.com/spatial-ucsb/ConceptsOfSpatialInformation, 29.03.2018
[7] https://github.com/sstuder/QuestionBasedSpatialComputing, 29.03.2018

assume a user to be a novice with little or no experience in spatial computing. The criteria were evaluated by an in-depth assessment and comparison of the operations used in the two analyses and the requisite adaptations of the core computations.

4 Results and evaluation of the analysis based on the language for spatial computing

4.1 Conventional analysis and analysis using the language for spatial computing

Figure 3 shows the conventional ArcPy script of the urban water analysis (left) and the Python script of the analysis based on the core concepts (right). The analyses will be discussed in detail in the next section.

While the available core concepts were sufficient for the case study, the existing core computations currently available in the Python library of Vahedi et al. (2016) have been extended. The four core computations shown in Table 3 were added to the core concept object.

Table 3: New implemented core computations (notation as in Kuhn and Ballatore (2015)).

CC	Operator: input parameters → output type	Comments
Object	restrictDomain: object x object → object	Restrict an object to the extent of another object
	get: object x (object → value) → value	Get the value of a property of the object
	addProperty: object x field → value	Add the value of a field as an attribute to the object
	withProperty: object x sql → object	Select object using an SQL expression

The *get*-method was mentioned in (Kuhn & Ballatore, 2015) but has not yet been implemented. The function *makeObject* already existed, but was modified within this contribution to make objects iterable and thus allow parts of an object to be treated as objects, as proposed by Kuhn & Ballatore (2015).

Additionally, we introduced 'helper methods'. We implemented the two helper methods, *save* and *show*. Helper methods are fundamental methods that are indispensable for coding but are not a core computation. The *save* method enables the saving of interim and final results to a file. In earlier ArcPy-based implementations of the language for spatial computing, the output of each ArcPy function was saved. In a sequence of questions within a script, the permanent storage of interim results is an undesired effect. The *show* method displays an attribute table of a temporary or a permanent object in the command-line interface and thus simplifies the handling.

Conventional analysis

```
# load input data
area = 'C:/area.shp'
dem = 'C:/dem.tif'
waterPoint = 'C:/waterPoints.shp'
building = 'C:/buildings.shp'
```

```
# set parameters
distance = 50
elevation = 3
```

```
# select water points within area
waterPoint_inArea = SelectLayerByLocation_management(waterPoint, 'INTERSECT', area)

# select buildings within area
building_inArea = SelectLayerByLocation_management(building, 'INTERSECT', area)

# elevation of waterPoints
waterPoint_elev = ExtractValuesToPoints(waterPoint_inArea, dem, 'in_memory/wp_elev')

# calculate elevation of buildings (using centroid)
building_point = FeatureToPoint_management(building_inArea, 'in_memory/building_point', 'CENTROID')
building_pt_elev = ExtractValuesToPoints(building_point, dem, 'in_memory/building_pt_elev')
building_elevation = CopyFeatures_management(building_inArea, 'in_memory/building_elevation')
JoinField_management(building_elevation, 'FID', building_pt_elev, 'FID', 'RASTERVALU')

cursor = SearchCursor(waterPoint_elev, ['OID@', 'SHAPE@', 'RASTERVALU'])
for wp in cursor:
   geom = wp[1]

   # select buildings within distance
   D = str(distance) + ' Meters'
   inDistance = SelectLayerByLocation_management(building_elevation, 'WITHIN_A_DISTANCE', geom, D)

   # select buildings within elevation
   sql = 'RASTERVALU >= ' + str(wp [2] - elevation) + ' AND ' +
   'RASTERVALU <= ' + str(wp [2] + elevation)
   WDWE = SelectLayerByAttribute_management(inDistance, 'SUBSET_SELECTION', sql)
```

```
   #  save output
   CopyFeatures_management(WDWE, 'C:/WDWE_' + str(wp [0]) + '.shp')
```

Core Concepts

```
# load input data
area = makeObject('C:/area.shp')
dem = makeField('C:/dem.tif').restrictDomain(area, 'inside')
waterPoint = makeObject('C:/waterPoints.shp').restrictDomain(area, 'inside')
building = makeObject('C:/buildings.shp').restrictDomain(area, 'inside')
```

```
# set parameters
distance = 50
elevation = 3
```

```
# Question 1: What are the elevations of the water points?
waterPoint_elev = waterPoint.addProperty(dem)

# Question 2: What are the elevations of the buildings?
building_elev = building.addProperty(dem)

for wp in waterPoint_elev:

   # Question 3: Which buildings are within the distance of the water point?
   wp_buffer = wp.buffer(distance, 'Meters')
   buildings_in_d = building_elev.restrictDomain(wp_buffer, 'inside')

   # Question 4: Which buildings are within the elevation parameter of the  water point?
   wpElev = wp.get('RASTERVALU')
   sql = 'RASTERVALU >= ' + str(wpElev - elevation) + ' AND ' +
   'RASTERVALU <= ' + str(wpElev + elevation)
   WDWE = buildings_in_d.withProperty(sql)
```

```
   # save output
   id = wp.get('FID')
   WDWE.save('C:/out', 'WDWE_' + str(id), '.shp')
```

Figure 3: Conventional analysis (top) and the analysis using the language for spatial computing (bottom).

4.2 Assessing the analysis conducted using the language for spatial computing

This section provides the evaluation of the analysis conducted using the language for spatial computing based on the criteria introduced in Section 3.2.

Question-based

With regard to the question-based criterion, we can say that the analysis using the core concepts answers specific questions, is declarative instead of procedural, and is more goal-oriented. This conclusion is based on the following observations.

For the analysis using the language for spatial computing, the main question was broken down into four sub-questions, each of which could be answered by performing one or two computations. This implies that a specific question can be answered using the core computations (Table 4). It gives the core computations a more declarative and less procedural character and brings the analysis closer to achieving its objective. The two analyses differ in answering the second question in particular: 'What are the elevations of the buildings? (see Table 4). A series of four computations (nos. 4–7 in Table 4) were needed in the conventional approach to add the property 'height' to the buildings. With the core computations, the user can simply ask for a property of an object using one single goal-oriented computation, *addProperty*; the procedure to calculate the buildings' heights is hidden in the library. *addProperty* directly uses the centroids of polygons as input and thus anticipates a decision. Whether this is user-friendly can be questioned. But with the procedure in the conventional analysis, the content of the question can easily be lost sight of. This is a prime example which shows that by using the core-concepts approach, the user can concentrate on answering a question instead of stringing together procedural computations.

The (for the user) abstract *SearchCursor* function in the conventional analysis that allows the iteration through each feature is also hidden in the library; the *searchCursor* function is implemented directly within the *makeObject* function (see Section 4.1). Thus, objects of the core computations are iterable without the user having to perform a computation for which there is no question.

Table 4: Answering sub-questions using the core concepts (left) and conventional analysis (right).

Sub-question	No.	Core Concepts	No.	Conventional Analysis
			1	SelectLayerByLocation_management()
			2	SelectLayerByLocation_management()
1) What are the elevations of the water points?	*1*	addProperty()	3	ExtractValuesToPoints()
2) What are the elevations of the buildings?	*2*	addProperty()	*4* *5* *6* 7	FeatureToPoints_management() ExtractValuesToPoints() CopyFeatures_management() JoinField_management()
Iteration	for wp in waterPoints		8	for wp in SearchCursor()
3) Which buildings are within the distance parameter of a water point?	*3* *4*	buffer() restrictDomain()	9	SelectLayerByLocation_management()
4) Which buildings are within the elevation parameter?	*5* *6*	sql get() withProperty()	10	sql SelectLayerByAttribute_management()
Save output	*7* *8*	get() save()	11	CopyFeatures_management()

Computational steps

The conventional analysis includes 11 computations, whereas the implementation with the core concepts contains 8 computations (7 core computations plus the helper-method *save*) (see Table 4). Thus, the question-based approach is 27% shorter. Although the considerable reduction by 45% in Vahedi et al. (2016) could not be reached, a reduction in the number of computational steps was verified in the present case study. Procedural calculations and abstract computations such as the *SearchCursor* function are hidden in the backend of the language for spatial computing and lead to a reduction in the number of computational steps.

The core concepts do not necessarily reduce the number of computations. Sometimes several core computations need to be combined to achieve a result for which one specific function exists in ArcPy. For example, the ArcPy function *SelectLayerByLocation_management* in the conventional analysis selects features within a defined distance; using the core concepts, two core computations, *buffer* and *restrictDomain*, were combined.

Comprehensibility

By reading in spatial data as a core concept, the core computations that a user can apply are limited to a certain number. This makes it easy for the user to grasp, select and apply possible commands to query spatial data. In addition, the syntax of the language for spatial computing is concise, and the descriptive terms make it intuitive and thus easy for novices to understand and learn the language. According to Ihaka and Gentleman (1996), the syntax of a computer language is only superficial, but it determines the way in which users of the language express themselves. Thus, the core concepts do not constitute a new GIS, but with the language for spatial computing they offer a new *superficial* approach to how users perceive and query space.

Role of base language

In addition to the functions and methods implemented within the Python library CoreConcepts, a user could also use any Python syntax, such as conditional statements, enumerate items, import other libraries, and so on. If a question cannot be answered using the core computations, other libraries could be used by more advanced users. Embedding the language for spatial computing in Python creates many possibilities, but it also requires that users gain some basic knowledge of Python. As a minimum, a user needs to know how to write a value to a variable, how to type strings, use Python methods, create for-loops, and know the rules of indentations. Implementations in other languages such as Haskell, RDF or JavaScript require the same basic knowledge.

Role of underlying GIS

Underlying GIS have a significant influence on the implementation of the core concepts, as existing functions are directly linked to core computations. In our case study, the core computations depend on ArcPy peculiarities such as the function *ExtractValuesToPoints* behind the *addProperty* computation that generates an extra field named 'RASTERVALU', the temporary memory 'in_memory', or the SQL dialects used in ArcGIS[8]. Thus, characteristics of the underlying ArcPy library have an impact on the implementation of the core concepts and would be different in the context of another GIS. Additionally, as recognized by Müller (2015), there is the difficulty that GIS operations are not standardized, and depending on the GIS on which the core concepts are built, underlying operations do not perform in exactly the same manner.

Role of data property

In the conventional analysis, data layers are loaded into a GIS application. As many GIS operations are format-dependent, such data properties lead to unnecessary conversions (Vahedi et al., 2016). If the water points in the case study were stored in a text file and the area of interest were a land-use class in a classified raster, in the conventional analysis several conversion steps would be needed before the spatial analysis could be conducted on the data. The text file would be converted to a feature and the raster class to a polygon. Using the core-concepts approach, the user views data from the perspective of the core concepts and

[8] http://pro.arcgis.com/en/pro-app/help/mapping/navigation/write-a-query-in-the-query-builder.htm, 29.03.2018

chooses the most appropriate concept to compute on the data in order to answer the spatial question. Vahedi et al. (2016) stated that the data format should not be a limiting factor for the core computation to be performed. Therefore, the core concepts must be implemented in such a way that all data formats are readable for each core concept. For our example, this means that when importing a text file as an object, the columns for the coordinates are automatically requested. Or, if a classified raster is loaded as an object, the classes are converted to polygons in the backend.

5 Further observations on the language for spatial computing

With regard to further development of the language for spatial computing, several aspects have to be taken into account. According to Kuhn and Ballatore (2015), a core-concepts approach aims to reduce the complexity and number of spatial computations by defining a set of core computations for each core concept. These operators form the semantic primitives of the language for spatial computing, which can be combined to conduct more complex spatial analyses. Kuhn and Ballatore (2015) provided a set of 29 core computations. Vahedi et al. (2016) introduced three computations, and we also added three core computations to conduct the specific case studies. To date, just which criteria the core computations should relate to has not been defined. If the number of core computations can be arbitrarily increased, they will end up being confusingly numerous, and the goal of a simple approach to spatial analysis will be missed. The difficulty now is to provide a number of relevant and universally applicable core computations which, when combined, will offer the user a maximum number of analysis possibilities. Once this has been achieved, it will be worth defining the core computations normatively and thus making the geoprocessing functionalities semantically interoperable. This would be the first top-down approach in the field of geoinformation as stated in Kuhn and Ballatore (2015).

A comprehensive language for spatial computing must take into account interoperability concerning the syntactic, meta and semantic levels. This requires that the language for spatial computing can be used on different platforms, that documentation of what a geoprocessing operation does exists, and that consistent terminology for operations across platforms guarantees that exactly the same process is performed. Due to the lack of meaningful descriptions of what geoprocessing tools do with data, users are often forced to acquire knowledge through the backend systems. Consistent behaviour of spatial computations across all platforms is desirable from a user's perspective and could be guaranteed by a common standard (Müller, 2015). The implementation of a language for spatial computing is an ideal starting point for a standard for geoprocessing functionalities.

The implementation of the core concepts based on existing GIS applications will pose some challenges with regard to the semantic interoperability of these implementations. In the case study, ArcPy peculiarities like the 'in_memory' workspace or the automatic generation of an attribute name were integrated in the implementation of the language for spatial computing. Implementations of normatively defined functions based on existing GIS applications are challenging. We question whether it makes sense to implement the language for spatial

computing based on existing GIS applications, as proposed in previous work, and wonder whether it would not be better to build a new language from scratch.

The language for spatial computing is supposed to work with any data format, no matter whether the data is provided in a file-based format or as linked data. Additional implementations that test a complete list of different data formats are needed. These tests need to include how the core concepts deal with rasters loaded as objects, or vectors as fields, networks or as another core concept.

Within this contribution, helper methods were introduced. They do not perform spatial computations; nor do they belong to a core concept. In our opinion, helper methods such as *save*, *showAttributes*, *plotMap* or *delete* are absolutely necessary, as a user needs computations other than the core computations for handling the data and memory usage. A definitive list of helper methods needs to be drawn up.

6 Conclusions

Our replication of the approach of Vahedi et al. (2016) using a case study from the humanitarian field showed that the language for spatial computing reduces the number of computing steps required for the desired analysis. The evaluation indicates that the use of core concepts encourages the interdisciplinary use of spatial analysis among non-GIS specialists because of its comprehensibility and its question orientation. Based on the experience gained from the case study, we expect that it would be easier for the GIS section of the ICRC to describe how data can be queried using the language for spatial computing than to develop and distribute ArcGIS script-tools to their engineers.

The implementation of additional core computations that were required for the case study led to suggestions for the future implementation of the language for spatial computing. These suggestions include defining the required number of core computations, the addition of helper methods, and normative definitions of geoprocessing functionalities. The strong dependency of the core computations on the underlying GIS poses a challenge if a standardized language is targeted.

Making a universal language for spatial computing accessible to diverse user communities holds a great potential for spatial analyses to be increasingly taken into account in decisions, no matter the discipline.

References

Albrecht, J. (1989). Universal analytical GIS operations — a task-oriented systematization of data structure-independent GIS functionality. In Onsrud H., Craglia M. (eds) *Geographic information research: transatlantic perspectives.* (pp. 577–591). Taylor and Francis.

Bearman, N., Jones, N., André, I., Cachinho, H. A., & DeMers, M. (2016). The future role of GIS education in creating critical spatial thinkers. *Journal of Geography in Higher Education*, *40*(3), 394–408. https://doi.org/10.1080/03098265.2016.1144729

Cai, G., Sharma, R., MacEachren, A. M., & Brewer, I. (2006). Human-GIS interaction issues in crisis response. *International Journal of Risk Assessment and Management*, *6*(4/5/6), 388. https://doi.org/10.1504/IJRAM.2006.009538

Dale, N., & Walker, H. M. (1996). *Abstract Data Types: Specifications, Implementations, and Applications*. Jones & Bartlett Learning.

Goodchild, M. F., Yuan, M., & Cova, T. J. (2007). Towards a general theory of geographic representation in GIS. *International Journal of Geographical Information Science*, *21*(3), 239–260. https://doi.org/10.1080/13658810600965271

Hofer, B., & Scheider, S. (under review). Geospatial Information Online Processing. In *Manual of Digital Earth*.

ICRC. (2017). *Calculating Buildings Being Supplied by a Water Point*. Draft tutorial about Urban Water Toolbox, Geneva, Switzerland.

Ihaka and Gentleman (1996). R: A Language for Data Analysis and Graphics. Journal of Computational and Graphical Statististics, 5, 299-314.

Janelle, D. G., & Goodchild, M. F. (2011). Concepts, principles, tools, and challenges in spatially integrated social science. *The SAGE Handbook of GIS and Society*. Thousand Oaks, CA: SAGE, 27–45.

Kuhn, W. (2012). Core concepts of spatial information for transdisciplinary research. *International Journal of Geographical Information Science*, *26*(12), 2267–2276. https://doi.org/10.1080/13658816.2012.722637

Kuhn, W., & Ballatore, A. (2015). Designing a Language for Spatial Computing. In F. Bacao, M. Y. Santos, & M. Painho (Eds.), *AGILE 2015* (pp. 309–326). Springer International Publishing. https://doi.org/10.1007/978-3-319-16787-9_18

Müller, M. (2015). Hierarchical profiling of geoprocessing services. *Computers & Geosciences*, *82*, 68–77. https://doi.org/10.1016/j.cageo.2015.05.017

Ntozini, R., Marks, S. J., Mangwadu, G., Mbuya, M. N. N., Gerema, G., Mutasa, B., … Chikwavaire, V. F. (2015). Using Geographic Information Systems and Spatial Analysis Methods to Assess Household Water Access and Sanitation Coverage in the SHINE Trial. *Clinical Infectious Diseases*, *61*(suppl_7), S716–S725. https://doi.org/10.1093/cid/civ847

Nüst, D., Granell, C., Hofer, B., Konkol, M., Ostermann, F. O., Sileryte, R., & Cerutti, V. (2018). Reproducible research and GIScience: an evaluation using AGILE conference papers. PeerJ, 6, e5072. https://doi.org/10.7717/peerj.5072

Scheider, S., Ballatore, A., & Lemmens, R. (2018). Finding and sharing GIS methods based on the questions they answer. *International Journal of Digital Earth*, *0*(0), 1–20. https://doi.org/10.1080/17538947.2018.1470688

Vahedi, B., Kuhn, W., & Ballatore, A. (2016). Question-Based Spatial Computing—A Case Study. In *Geospatial Data in a Changing World* (pp. 37–50). https://doi.org/10.1007/978-3-319-33783-8_3