STANDARDS-BASED SERVICES FOR BIG SPATIO-TEMPORAL DATA

P. Baumann [a,*], V. Merticariu [b], A. Dumitru [b], D. Misev [b]

[a] Jacobs University, Germany, - p.baumann@jacobs-university.de
[b] rasdaman GmbH, Bremen, Germany – {merticariu,dumitru,misev}@rasdaman.com

KEY WORDS: Big Data, datacubes, coverages, WCS, WCPS, standards, OGC, rasdaman

ABSTRACT:

With the unprecedented availability of continuously updated measured and generated data there is an immense potential for getting new and timely insights - yet, the value is not fully leveraged as of today. The quest is up for high-level service interfaces for dissecting datasets and rejoining them with other datasets - ultimately, to allow users to ask "any question, anytime, on any size" enabling them to "build their own product on the go".
With OGC Coverages, a concrete, interoperable data model has been established which unifies n-D spatio-temporal regular and irregular grids, point clouds, and meshes. The Web Coverage Service (WCS) suite provides versatile streamlined coverage functionality ranging from simple access to flexible spatio-temporal analytics. Flexibility and scalability of the WCS suite has been demonstrated in practice through massive services run by large-scale data centers.
We present the current status in OGC Coverage data and service models, contrast them to related work, and describe a scalable implementation based on the rasdaman array engine.

1. INTRODUCTION

With the unprecedented increase of orbital sensor, in-situ measurement, and simulation data as well as their derived products there is an immense potential for getting new and timely insights - yet, the value is not fully leveraged as of today. Incidentally, such spatio-temporal sensor, image, simulation, and statistics data in practice typically constitute prime Big Data contributors. In view of such data "too big to transport" the quest is up for high-level service interfaces for dissecting datasets and rejoining them with other datasets - ultimately, to allow users to ask "any question, anytime, on any size" enabling them to "build their own product on the go".

The notion of coverages has proven instrumental in unifying regular and irregular grids, point clouds, and meshes so that such data can be accessed and processed through a simple, yet flexible and interoperable service paradigm. Complementing the (abstract) coverage model of ISO 19123 on which it is based, the (concrete) OGC coverage data and service model establishes verifiable interoperability. The OGC Web Coverage Service (WCS) comprises a modular suite for accessing large coverage assets. WCS Core provides simple data subsetting and encoding whereas extensions add optional service facets up to ad-hoc spatio-temporal filtering and processing on massive datacubes. The latter is accomplished by OGC's Big Earth Data query language, Web Coverage Processing Service (WCPS). By separating coverage data and service model, any service - such as WMS, WFS, SOS and WPS - can provide and consume coverages in addition to WCS, thereby enabling heterogeneous mashups with lossless information flow.

The critical role of coverages for spatio-temporal infrastructures is recognized far beyond OGC: a large, growing list of open-source and proprietary tools support WCS today. WCS/WCPS implementations host databases beyond 100 Terabyte, heading towards the Petabyte frontier, and WCPS queries have been distributed across more than 1,000 cloud nodes. This has prompted ISO and INSPIRE to also adopt the OGC coverage and WCS standards, which is currently under way. The standards observing group of the US Federal Geographic Data Committee (FGDC) sees coverage processing a la WCS/WCPS as a future "mandatory standard".

In this paper, we introduce the OGC coverage data and service model with an emphasis on practical aspects and illustrate how they enable high-performance, scalable implementations. Presentation will make use of online available services allowing participants to follow and recapitulate the topics addressed.

The remainder of this paper is organized as follows. In Section 2 we discuss the OGC Coverage model as a unifying “Big Data” paradigm. The WCS suite is presented in Section 3, followed by an overview of the OGC Core Reference Implementation, rasdaman, in Section 4. Related Work is discussed in Section 5, and in Section 6 findings are summarized.

2. THE WCS “BIG GEO DATA” STANDARDS SUITE

2.1 The Coverage Data Model

According to OGC and ISO, features form general spatio-temporal objects, of which coverages are a particular specialization [20][26]. Coverages are special in that they are “spatio-temporally varying” which means: at different positions they have different values. For example, a polygon feature bearing an attribute “Highway A1” will have this for every location along the polygon; a raster image, conversely, has individual values at each of its pixel positions. Obviously, this makes coverages massively more voluminous in comparison to features – in modern terminology, coverages usually constitute the “Big Data”.

Formally, coverages are defined as functions where the *domain* defines the locations (called *direct positions*) for which a coverage provides values and the *range* defines the set of values ass-

* Corresponding author

ociated with any direct position. This concept is general enough to describe multi-dimensional regular and irregular grids, point clouds, and general meshes. ISO 19123 (identical to OGC Abstract Topic 6) establishes the abstract model; this model is not yet interoperable per se, as it allows for manifold incompatible implementations. Therefore, this abstract model is complemented by a concrete implementation model, the OGC *Coverage Implementation Schema* (CIS) [8]. CIS is defined concise enough to allow conformance testing down to the level of single pixels, hence can be considered interoperable.

The OGC coverage model is derived from coverages as defined in GML 3.2.1 [30], but lifting the GML-based definitions to a conceptual model that is encoding independent[1]. Further, the GML 3.2.1 model turned out to be insufficient in semantics; to remedy this, the coverage standard extends the GML 3.2.1 model in a backwards compatible way by adding the concept of a grid cell ("pixel", "voxel") type. This definition, the so-called range type is adopted from SWE Common [32]. Range type information is not constrained to data types, such as int or float; rather, it comprehensively describes the semantics of range values through concepts identified by URL s (e.g., for radiance), units of measure (such as Watt per square centimetres), null values, and additional constraints. This way, coverages are informationally complete without any loss of semantics in processing pipelines even across different tools and standards.

In practice, applications often want to store specific metadata alongside with the coverage data. To support this, coverages contain a metadata compartment capable to hold any unspecified contents. By definition, these metadata will always be carried along during retrieval and transport of the coverage and, hence, never get lost.

This basic model of coverages – consisting of mainly the components domain set, range type, range set, and metadata – has served well for regular grids, but, due to historical reasons, did not have a convenient, flexible way of handling irregular grids. In CIS 1.1, therefore, a general, unified grid model has been established which allows expressing any kind of non-regular grid, thereby integrating, simplifying, and extending GML 3.2.1 and GML 3.3 attempts in the same direction2. The main difference is that coverages are not categorized along the complete grid, leading to cumbersome categories like Rectified Grid Coverage (with all axes regular) and Referenceable Grid Coverage (with at least one axis non-regular), but rather axis types are classified; axes of any type can be combined freely into grids, thereby achieving a general, yet concise description. The axis types foreseen in CIS 1.1 are, in ascending order of algorithmic complexity:

A index axis which is just Cartesian, without any units and a spacing of 1. It may represent any abstract or unknown measure. Formerly, Cartesian grids were sometimes called Image CRS, albeit without any concise definition (Figure 1 left top).

- A regular axis adheres to some reference system with a particular datum, axis direction, and unit of measure. Spacing of direct positions along such an axis is equidistant, therefore it is sufficient to store this stepping, i.e.: resolution (Figure 1 left).
- An *irregular axis* is as before, but with a stepping of direct position locations that can vary. Hence, all these positions need to be stored explicitly (Figure 1 center bottom).
- Grids where the direct positions are not lined up along straight lines are called *warped* or *displacement grids* (Figure 1 center top). As not all axes of a grid need to participate in such a warping, the set of axes participating is called a *warped axis nest*. Note that, while coordinates are not aligned any longer (and can even be embedded into some higher-dimensional space), they still topologically resemble a grid: every inner grid point has exactly two direct neighbours along any axis. Obviously storing such information needs a coordinate tuple for each value of the range set – this is where the domain set can occupy a volume similar to the range set (which often – wrongly – has been considered the only "Big Data" part of a coverage).
- The ultimate freedom in defining a grid is given by *transformation* grids. The coverage contains only some input ingredients for some externally defined black box algorithm generating the direct positions. Currently, one such method is standardized which is based on SensorML 2 [11].

Rectified Grid Coverages as per GML 3.2.1 resemble grids where all axes are regular. Non-regular grids, sumamrized as Referenceable Grid Coverages, are not defined unambiguously in GML 3.2.1, therefore GML 3.3 (which is an allowed option in CIS 1.1) distinguishes between *Referenceable Grid By Vector* (corresponding to irregular axes), *Referenceable Grid By Array* (resembling displacement grids), and *Referenceable Grid By Transformation* (resembling transformation axes). This ignores mixed grids which frequently occur in practice (cf. Figure 1 right) – for example, a satellite image timeseries may have orthorectified (i.e., regular) axes in Lat and Long while images are taken at irregular times, yielding an irregular time axis. CIS 1.1 allows any combination of axis types.

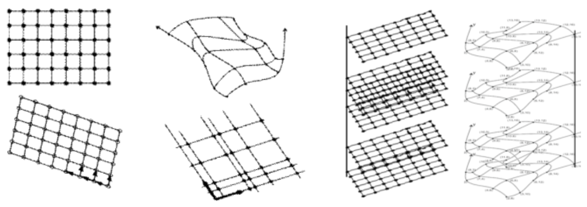

Figure 1. Sample coverage grid types [8]

Every axis has a definition, expressed by a URL pointing to a resource defining it; this can be a single axis definition, or an axis being defined as part of a Coordinate Reference Systems (CRS). The canonical OGC resolver for CRSs is accessible under http://www.opengis.net/def/crs; by replacing the crs part accordingly, definitions of axes, datums, etc. can be retrieved. Among others, this open-source resolver [24] offers CRSs as defined by EPSG, but also time and index axis definitions. Through a specific URL pattern, CRSs and axes can be combined to shape new multi-dimensional CRSs. The following URL below establishes a CRS for 3-D image timeseries with axes *Lat* and *Long* (as defined by EPSG:4326), and *ansiDate*:

[1] CIS was formerly known as "GML 3.2.1 Implementation Schema – Coverages" (GMLCOV) [6]; as this title caused manifold confusions (such as being just a GML encoding) its title was modified to CIS in Spring 2015.

[2] In 2016, a poll within OGC revealed that there is no implementation of GML 3.3 irregular grids existing.

http://www.opengis.net/def/crs-compound?
1=http://www.opengis.net/def/crs/EPSG/0/4326&
2=http://www.opengis.net/def/crs/OGC/0/ansiDate

Encoding of coverages is possible in a variety of ways, yet controlled for interoperability. One approach is to use general-purpose encodings like XML or JSON. On the upside, such encodings are able to represent all coverage information, both generator and parser tools are readily available making them often convenient for use (such as using JSON when deriving 1-D diagram or 2-D image coverages for display in a Web browser); on the downside, such (typically ASCII-based formats tend to lead to rather voluminous representations, hence do not scale with large coverages. In such cases, efficient binary encodings like TIFF, NetCDF, or JPEG2000 provide an alternative – albeit at the cost of not retaining all information. Combining the best of both approaches – information completeness and volume efficiency – is possible through container formats which allow splitting coverages into sub-parts each of which gets encoded individually. For example, the coverage domain set, range type, metadata, etc. might get encoded in XML or JSON while the "pixel payload" – the range set – gets encoded in NetCDF. Suitable as a container format are formats like multipart MIME (such as used for email attachments), zip, GMLJP2 [14], etc.

So far we have assumed that coverages are stored along their conceptual definition, with separate components for domain and range set. Due to manifold stakeholder requests this has been extended in version 1.1 to allow a fine-grain partitioning of coverages. Instead of the domain/range representation, a coverage may be recursively composed of sub-coverages. A second option, tuned towards timeseries generation, consists of organizing a coverage into a sequence of position/value pairs.

Note though, that both features should not be used naively as a storage organization in a server; it is a strength of the coverage model that it allows to organize data efficiently towards particular service functionality and access patterns while retaining all information and semantics. For example, timeseries analysis on an x/y/t coverage may suffer from inadequate performance if organized into horizontal slices (as done traditionally). Servers supporting adaptive tiling can be optimized towards any particular access pattern and, hence, are known to convey substantially better performance [22][15].

This summarizes our overview on CIS 1.1. We have discussed various facets which are orthogonal in the standard, i.e., can be combined. Technically, in the specification this is achieved by packaging functionality into conformance classes, some being dependent on others. Figure 2 shows all conformance classes and their dependencies synoptically.

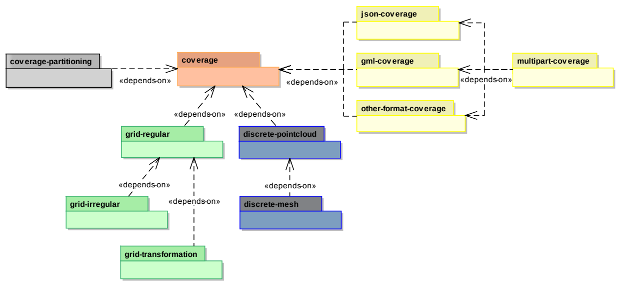

Figure 2. Coverage Implementation Schema (CIS) 1.1 conformance classes [8]

2.2 The Coverage Service Model

This section addresses service interfaces for flexible access to and processing of coverages. Being specializations of features, coverages on principle can be processed and served through any OGC-based service knowing features, such as WMS, WFS, WPS, and SOS. However, as these do not foresee coverage-specific operations the main functionality offered is download of the complete object, which often is not feasible. This is where the dedicated coverage services come in, the *Web Coverage Service* (WCS) suite [7]with its coverage query language, *Web Coverage Processing Service* (WCPS) [5].

WCS follows a modular approach: centered around a simple core with minimal functionality every conformance claiming implementation must support, a series of extensions is available adding functionality facets. Application profiles streamline and specialise WCS for particular use cases (such as remote sensing) by bundling core and selected extensions and adding further rules and functionality.

WCS Core is tentatively basic: its *GetCoverage* request allows access to a coverage or a subset thereof, and encoding into some user selected data format. Along each axis, subsetting can be either *trimming* (Figure 3 left), which shrinks extent while retaining dimension, or *slicing* (Figure 3 right), which cuts out a hyperplane at the given point, thereby reducing dimension. Trimming and slicing can be mixed arbitrarily on all axes. All is guaranteed to deliver the original, unchanged data (unless a lossy format encoding is chosen, of course); OGC conformance tests allow every implementation to evaluate itself on this [27].

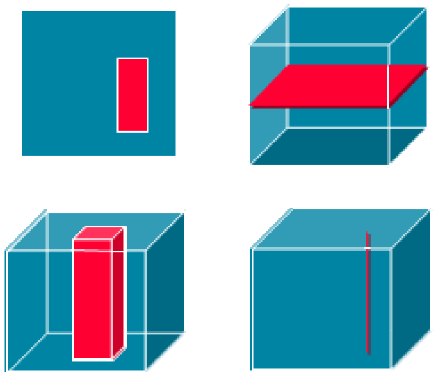

Figure 3. WCS subsetting: trimming (left) and slicing (right) (source: Wikipedia [35])

WCS extensions are grouped into several categories, currently Data Model, Service, Protocol Binding, and Usability extensions. Only service and protocol extensions are available at the time of this writing, so we will discuss only these.

Service extensions add functionality facets (Figure 4): Range subsetting allows extraction of particular components ("bands", "variables") from composite range values. Scaling reduces resolution of gridded coverages. CRS transformation reprojects a coverage into a target CRS prior to delivery. Interpolation gives control over any server-side interpolation applied during request processing. WCS-T (for "Transaction") provides maintenance capabilities for creating, deleting, and updating coverage offerings on a server; particularly important is the ability to update parts of a coverage which enables, e.g., piecewise creation of image mosaics or timeseries. The WCS Processing extension will be discussed in the next section.

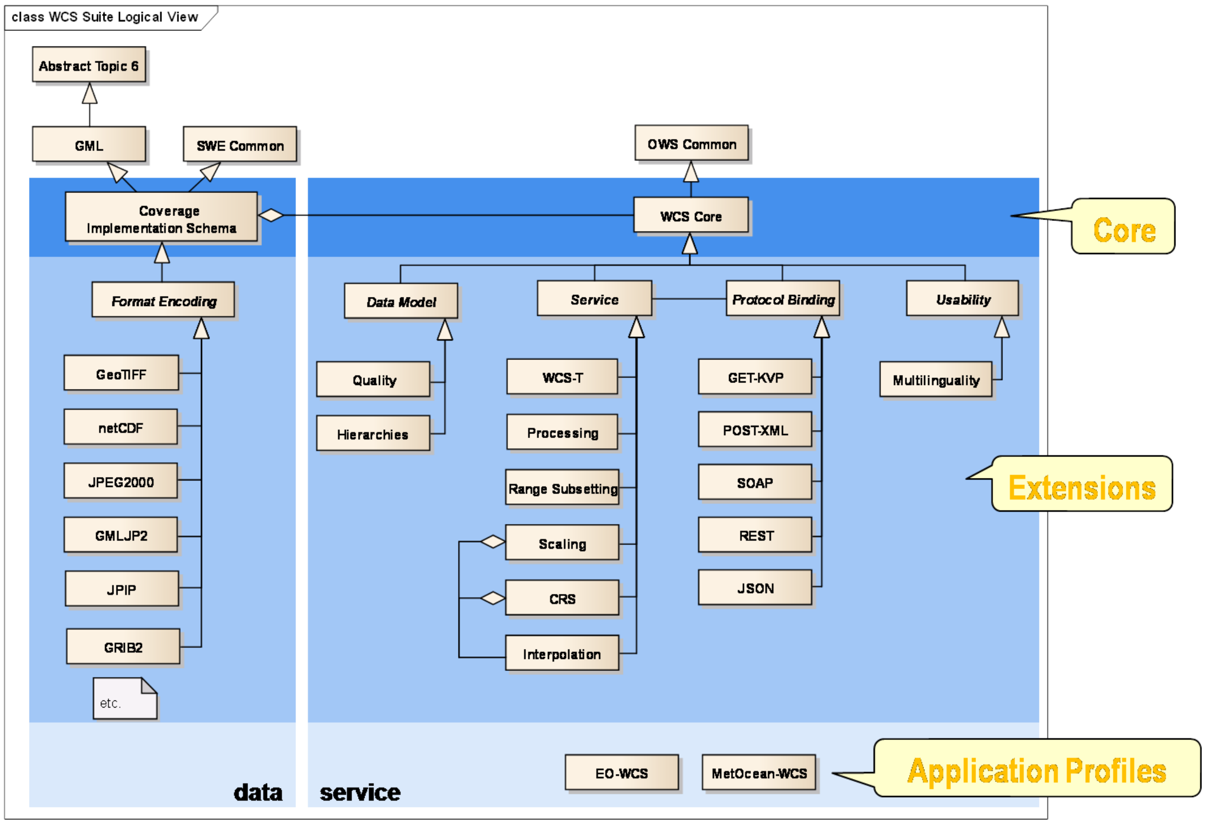

Figure 4. WCS modular data and service model
(source: Wikipedia [35])

Protocol extensions allow requests to be expressed in all common techniques, including GET/KVP, XML/POST, SOAP, and (on draft level) REST. This way, request functionality and encoding are separated orthogonally.

This variety of possible server implementations opens up a wide space for vendors to provide solutions, while retaining inter-operability. However, there is the risk that a client requests functionality that the particular server addressed does not offer. To this end, the canonical *GetCapabilities* request (as defined in OWS Common) delivers additional service quality information:

- identifiers (URLs, actually) of all extensions supported
- identifiers of all CRSs supported
- identifiers of all data formats supported

Altogether, the conceptual model of a WCS is an offering as de-scribed in Figure 5. Service metadata are at the root (which can be retrieved through *GetCapabilities*), underneath the set of coverages offered. Each coverage can be ornamented individ-ually with coverage-specific service metadata; this is mainly re-served as hooks for future extensions that might eventually need this. Under preparation is a search and discovery extension which, based on XPath, allows extracting any information stored in a server's offering. A few examples may illustrate this:

- "All encoding formats supported":
  //formatSupported
- "All WCS extensions supported":
  //Profile
- "Identifiers of all timeseries":
  //coverage[*envelope@dimension=3*]*/@id*

Note that the server may deny requests leading to excessive data sets, such as "All pixels", denoted as *//rangeSet*.

2.3 Coverage Analytics

We now turn to WCPS, that WCS extension which has been postponed in our earlier overview. In a nutshell, WCPS is a query language for massive multi-dimensional raster objects with builtin geo semantics based on the coverage model [5][4].

Array operations include local, focal, zonal, and global operat-ions as per the classification by Tomlin [34]. Trimming and slicing is supported like in WCS Core. Arithmetic, Boolean, ex-ponential, logarithmic, and trigonometric functions can be app-lied to coverage cells simultaneously. Aggregation functions all-ow summarization along space and time. Data fusion is support-ed by combining any number of coverages to derive new results. In summary, the language allows formulating arbitrarily com-plex expressions for signal, image, and statistical processing.

As today's metadata often are organized into hierarchical data records represented in XML or JSON, WCPS syntax lends itself towards a hierarchical query language, XQuery. Instead of a systematic, detailed overview, which can be found in [5], we explain the flavour of the language by way of an example: "from MODIS scenes M1, M2, M3, deliver the difference between red and near-infrared (nir) bands, encoded in TIFF; but only those where nir exceeds 127 in some pixel":

```
for $c in ( M1, M2, M3 )
where some( $c.nir > 127 )
return  encode( $c.red - $c.nir, "image/tiff" )
```

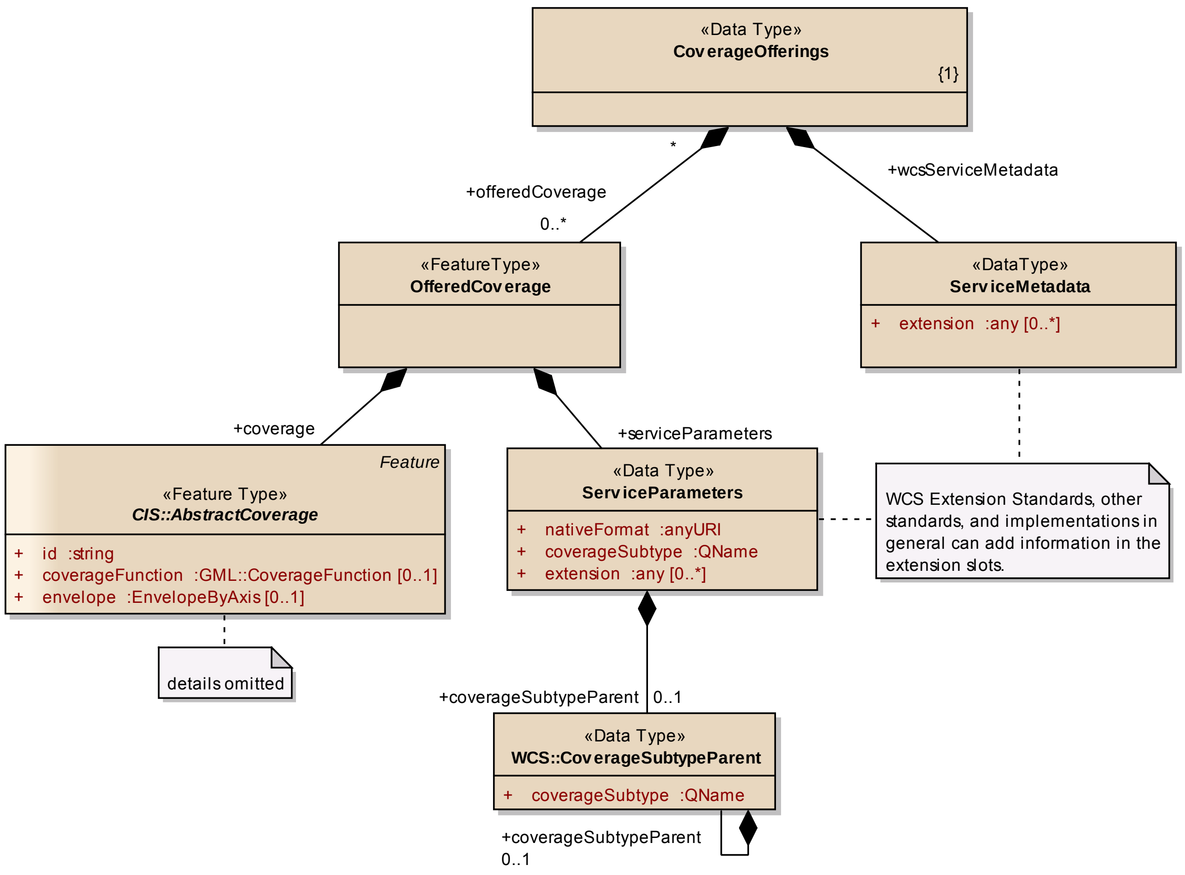

Figure 5. Conceptual model of a WCS offering [7]

The next query constructs a timeseries summary from a 3-D x/y/t datacube by conflating the history of each pixel into an average value; hence, the result is a 2-D x/y image which can be conveniently shipped in PNG:

```
for $c in ( TimeseriesDatacube )
return  encode( coverage Average
                    over $x in $c.domain[“Lat”],
                         $y in $c.domain[“Long”]
                    values avg( $c[ Lat($x), Long ($y) ],
                   “image/png“
          )
```

In this request, a new coverage *Average* is created which has the same Lat and Long extent as *TimeseriesDatacube* (cf. *over* clause). The value of each pixel is given by the average of all cell values along the complete timelines, for each Lat/Long position individually (cf. *values* clause). Figure 6 shows some WCPS query results obtained through rasdaman (see next section). Note the WMS screenshot (second from right) which consists of four layers: a greyscale orthoimage, water areas and water layers, and a classified elevation model. This overlay has been derived through a single query, as rasdaman internally maps WMS requests to raster queries. Overlaying in the server increases efficiency and, hence, is advantageous for the overall response times.

Technically, an additional request type, *ProcessCoverages*, accepts a query string and returns a set of coverages or scalars. As with WCS in general, several protocols are available for client/server communication.

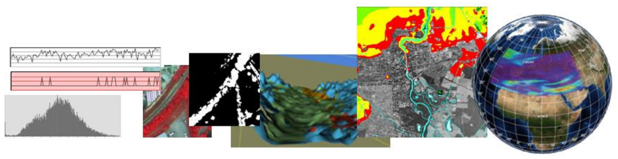

Figure 6. Sample WCPS retrieval results (source: EarthServer)

Currently WCPS 2 is under preparation which is supposed to offer a more general and powerful, yet easier to use programming model, in particular: tight data / metadata integration in queries [21].

3. IMPLEMENTATION

The most comprehensive implementation of WCPS (and reference implementation) is rasdaman (“raster data manager”), a highly scalable array analytics engine. It offers an array query language with a formal algebraic semantics underneath [2][3] which effectively has served as blueprint for the forthcoming ISO SQL/MDA (Multi-Dimensional Arrays) standard [23].

In the server, arrays are partitioned into so-called tiles which form the unit of access. Partitioning can be streamlined to optimally support any given workload of access patterns. Based on this approach, databases exceeding 100 TB are running rasdaman on spatio-temporal datacubes [9].

Incoming queries undergo a series of optimizations and can be processed in distributed environments federating laptops, clouds, and data center. Single queries have successfully been split across more than 1,000 cloud nodes [15].

Figure 7 shows the overall system architecture of rasdaman. The central workhorse, rasserver, operates in a multi-parallel fashion without a single point of failure (such as Hadoop has, for example). Array partitions can sit in some database or directly in the file system. As an additional option, external pre-existing archives can be included into queries. In this case, arrays are built from the files found there, whereby conventions like file and directory naming are considered during incremental registration.

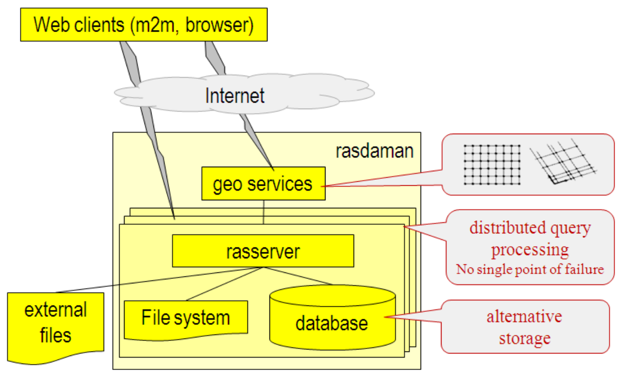

Figure 7. rasdaman federation architecture

As a domain-agnostic array engine, rasdaman operates on bare arrays. Geo semantics is added through an additional layer which implements coverages and offers them through OGC WMS, WCS, and WCPS interfaces.

In the EarthServer initiative (Figure 8), large-scale data centers use rasdaman for establishing a planetary-scale datacube federation [16][9][10]. The European Centre for Medium-Range Weather Forecast (ECMWF) maintains a climate data archive amounting to 87 PB. The European Space Agency (ESA) has more than 100 TB online at this time of writing, increasingly adding Sentinel data. This service is administrated by MEEO which adds in their Multi-Sensor Evolution Analysis System (MEA) as a climate data management platform operating with rasdaman as backend. Plymouth Marine Laboratory (PML) has entered the federation with ocean colour analysis [17] based on Landsat, MODIS, SEAWIFS, etc., and increasingly Sentinel. National Computational Infrastructure (NCI) Australia contributes reprocessed Landsat data, among others. Currently, datasets are being extended to cross the Petabyte barrier within the next months.

In 2016, datacube fusion (i.e., array joins) have been shown involving a 4-D climate datacube hosted by ECMWF close to London, UK and a Landsat image datacube hosted by NCI at Canberra, Australia (Figure 9). Additionally, the PlanetServer data hub offers data on Mars, Moon, and soon Vesta and more planetary bodies [29][13].

According to independent reviewers, EarthServer will "significantly transform the way that scientists in different areas of Earth Science will be able to access and use data in a way that hitherto was not possible". They attested "proven evidence" that rasdaman will “significantly transform [how to] access and use data“.

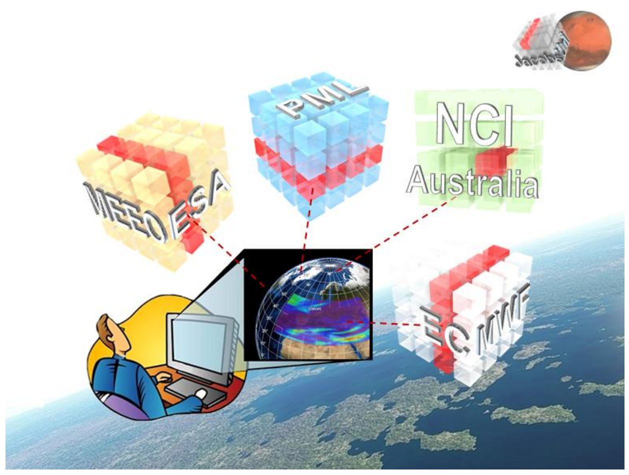

Figure 8. Intercontinental datacube mix and match in the EarthServer initiative (source: EarthServer)

Importantly, datacube languages like WCPS and Array SQL are not envisioned as user interfaces. Similar to WCS, they rather constitute a convenient machine-to-machine interface between servers and client frontends like visual globes or desktop GISs (Figure 10), but in particular in the middle of automated processing chains where there is no human interaction and control. In particular the latter use case is an important scenario for the “Big Data” services under discussion here. Generally speaking, array interfaces should at best be visible to experts as “power tools” – but even then well-known environments like python and R offer, once coming with a tightly integrated rasdaman backend, a more convenient experience as experts do not need to leave their comfort zone of tool skills.

4. RELATED WORK

In this section, we assess the coverage data and service model against related approaches.

4.1 Data Models

ISO 19123 / OGC Abstract Topic 6 defines an abstract coverage model. This model tentatively is kept general, hence incompatible implementations are possible. The OGC Coverage Implementation Schema (we recall, it was nicknamed GMLCOV, now it is CIS) complements this with a concrete, conformance testable coverage structure definition which allows for interoperable implementations. This has been acknowledged by ISO where CIS currently is under adoption as ISO 19123-2. ISO 19123 is scheduled for revision, after which it will become 19123-1 forming the abstract companion standard to concrete 19123-2.

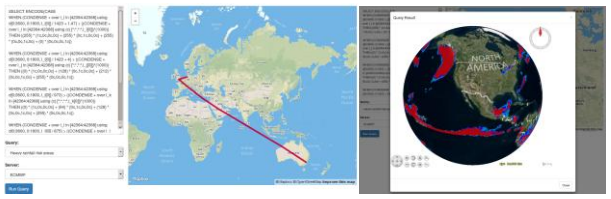

Figure 9. Datacube fusion between ECMWF / UK and NCI Australia: query distribution path (left) and result visualization (right) (source: EarthServer)

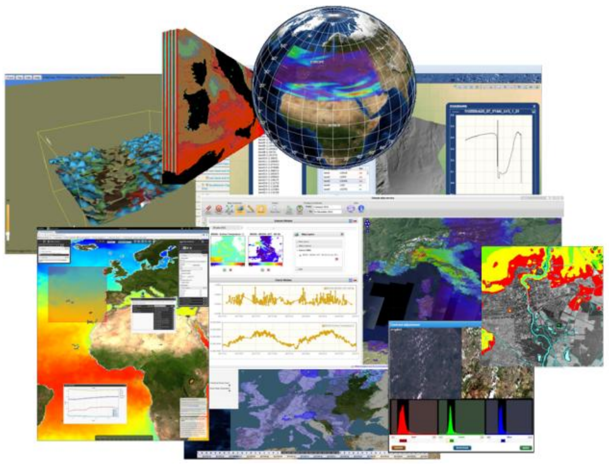

Figure 10. Screenshots of domain and task specific client frontends accessing rasdaman via WCPS (source: EarthServer)

GML 3.2.1 [30], among others, contains a concrete coverage model encoded in GML. Concepts and XML Schema have been adopted by the OGC WCS standards working group in 2010 through 2012 in an effort to harmonize and unify coverage definitions all across OGC. Although several conceptual weaknesses were spotted in GML (such as an underspecification of grids), care was taken in all necessary adaptations to keep changes backwards compatible. The main outside visible changes ultimately were addition of range type and metadata, as discussed earlier. However, there were several additions. In the model, support for interpolation has been added. Importantly, coverage representation in GMLCOV/CIS 1.0 has been liberated from GML and can use any suitable encoding. CIS 1.1 extends this further by allowing more partitioning variation with more container formats.

GML 3.3 [31] adds several grid types to GML 3.2.1, crafted independently from the OGC Coverage model. Further, due to the OGC modular specification rules and XML namespace rules these definitions are not automatically available in GMLCOV/CIS 1.0. Further, GML 3.3 grid types resemble only special cases omitting, for example, combinations of regular and irregular axes in the same datacube. The CIS 1.1 model encompasses and generalizes GML 3.3. In the CIS 1.1 XML encoding, the GML 3.3 schema is included.

No operational implementation of coverages is known which relies on GML coverages, neither for GML 3.2.1 nor for GML 3.3. Actually, a survey conducted in 2016 across OGC has revealed that there is no implementation of GML 3.3 known at all. OGC Coverages, on the other hand, have been implemented widely by both open source and proprietary tool developers and are in successful use since many years, on databases beyond 100 TB [9].

WaterML 2, coming from a 1-D timeseries representation, has extended this to consider images in place of the scalar measurements. While this is a suitable approach for upstream collection of timeslices it will incur performance penalties for access and evaluation (Figure 11).

In passing we observe that WaterML, incompatible with OGC Coverages while both are derived from ISO 19123, underlines that the abstract ISO model does not enforce interoperable concretizations.

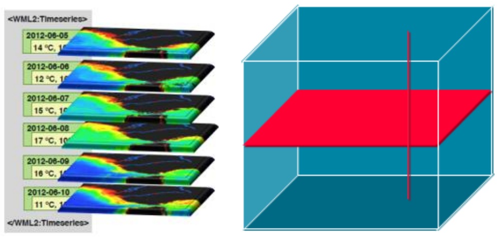

Figure 11. Sliced datacube representation in WaterML (let) vs. direction-invariant coverage model (right)

TimeseriesML is another, more recent a standard dedicated to timeseries. It is a further development of WaterML with likewise extensive metadata modelling. Also like WaterML it includes coverage-type structures. However, instead of establishing own structures it references coverages where applicable. This appears as a good example of standards generating added value in combination.

Generally speaking, TimeseriesML, etc. represent domain-specific standards for which the OGC Coverage Implementation Schema establishes a domain-neutral basic data structure which can be used whenever a coverage-like structure occurs; such standards, while retaining interoperability by using the common coverage model, will likely extend coverages with domain specific metadata, such as done in TimerseriesML.

W3C has issued a datacube model based on RDF [33]. While it is appealing to embed coverages into SPARQL queries this datacube appears less efficient in that it needs six RDF triples for each coverage cell. This makes RDF datacubes infeasible for scalable coverage handling. An alternative modelling, named SciSPARQL, based not on an RDF representation of arrays but on a model integration of SPARQL with OGC Coverages, has been published in [1] where promising prototype experiences are reported.

4.2 Service Models

OGC Sensor Observation Service (SOS) establishes Web service interfaces for retrieving "observations, sensor metadata, as well as representations of observed features" [12]. Like WCS, it allows for inserting new items into a service's offerings, however, without the ability to partially establish or update (potentially large) coverages. As opposed to WCS, there is no RESTful binding, just GET/KVP and SOAP. Notably, SOS and WCS are not competing: While SOS excels in collecting and unifying upstream data from virtually any sensor, WCS offers versatile functionality for discovering, filtering, extracting, and processing on spatio-temporal sensor, image, timeseries, simulation, and statistics data (Figure 12).

OGC Web Processing Service (WPS) [25] is a service specification which essentially transposes Remote Procedure Calls (RPCs) into Web world. The concept of RPCs is implemented and in use in manifold ways, e.g., in CORBA [28], Java Remote Method Invocation (RMI) [36], SOAP [19], and recently Google Web Toolkit [18]. WPS provides synchronous and asynchronous Web service invocations based on XML schemas where function signatures (i.e., names and parameterization) are defined. Functionality offered this way needs to be configured by an administrator to become available.

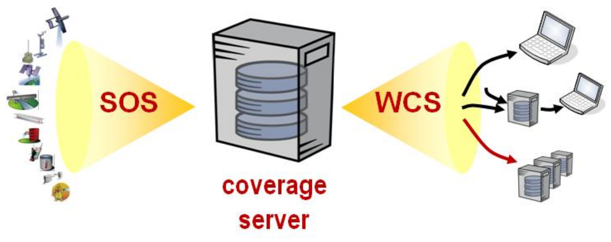

Figure 12. Complementary purposes of SOS and WCS: upstream sensor collection (left) and downstream download, processing, and visualization (right)

WPS and WCPS both allow to invoke server-side processing via Web interfaces. The difference between both is in the API approach of WPS versus the language approach of WCPS which has two important consequences:

- WCPS is (tentatively) limited in its expressiveness whereas WPS can offer any kind of functionality. This makes WCPS amenable to versatile data analytics and avoids certain denial of service attacks, but some algorithmical tasks such as complex coded simulations are not feasible with WCPS.
- WCPS allows any query, anytime whereas WPS processes need to be defined and configured by an administrator first.
- WCPS requests have a well-defined semantics (which allows automatic query generation by tools) whereas WPS processes are described only by the function signature and human readable text in title and abstract (Figure 13). This is not amenable to automatic orchestration and interoperable semantics.

In summary, these standards address different purposes:

- WCS provides simple and flexible access to coverages, with clearly defined, interoperable semantics;
- WCPS provides ad-hoc analytics on coverages, with clearly defined, interoperable semantics;
- WPS provides general-purpose processing (which can include coverages), without concise interoperability.

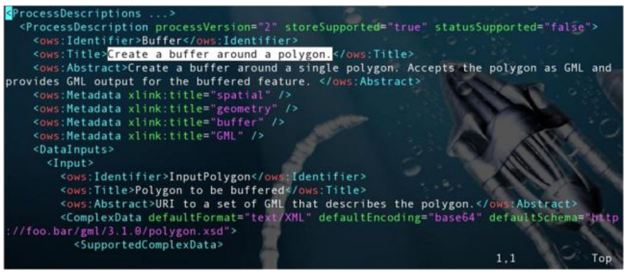
```
<ProcessDescriptions ...>
  <ProcessDescription processVersion="2" storeSupported="true" statusSupported="false">
    <ows:Identifier>Buffer</ows:Identifier>
    <ows:Title>Create a buffer around a polygon.</ows:Title>
    <ows:Abstract>Create a buffer around a single polygon. Accepts the polygon as GML and
provides GML output for the buffered feature. </ows:Abstract>
    <ows:Metadata xlink:title="spatial" />
    <ows:Metadata xlink:title="geometry" />
    <ows:Metadata xlink:title="buffer" />
    <ows:Metadata xlink:title="GML" />
    <DataInputs>
      <Input>
        <ows:Identifier>InputPolygon</ows:Identifier>
        <ows:Title>Polygon to be buffered</ows:Title>
        <ows:Abstract>URI to a set of GML that describes the polygon.</ows:Abstract>
        <ComplexData defaultFormat="text/XML" defaultEncoding="base64" defaultSchema="http
://foo.bar/gml/3.1.0/polygon.xsd">
          <SupportedComplexData>
                                                                          1,1          Top
```

Figure 13. Sample WPS process definition, with semantics explained in title and abstract

5. CONCLUSION

We presented a unified model for massive spatio-temporal data sets, such as ortho imagery, elevation data, image timeseries, climate data, and many more. OGC Coverage data and services have seen massive uptake – the major open source as well as proprietary tools today support it. WCS/WCPS-based coverage services are in use on large scale deployments; currently, the next step is being taken by addressing fusion of datacubes stored in different data centers. Goal is to achieve a unique mashup or federation of data centers resulting in a single information space supported by standards-conformant services.

At the time of this writing, CIS 1.1 has undergone critical assessment by various stakeholders, and is being implemented in parallel. The specification is in the final stage of the OGC adoption process, after which it is planned for adoption by ISO TC211 forming 19123-2. This concrete implementation model will be paired by a modernized version of the abstract model in 19123, planned to be renamed to 19123-1. Subsequently, ISO adoption of WCS 2 is foreseen.

Meanwhile WCS 2 is also advanced in the adoption process of INSPIRE, the European legal framework for a common spatial data infrastructure.

Altogether, coverage data and service models represent a major step forward towards a common information space for “Big Geo Data” enabling unified, interoperable access, processing, and fusion of spatio-temporal data.

Future work includes enhancing WCS with further extensions deemed necessary, finalizing WCPS 2.0, finalizing CIS 1.1 implementation in rasdaman and establishing a conformance test suite for the new standard.

ACKNOWLEDGEMENTS

This research is being co-funded by the European Commission through project EarthServer-2.

REFERENCES

[1] Andrejev, A., Baumann, P., Misev, D., Risch, T., 2015. Spatio-Temporal Gridded Data Processing on the Semantic Web. 2015 IEEE Intl. Conf. on Data Science and Data Intensive Systems (DSDIS 2015), Sydney, Australia.

[2] Baumann, P., 1994. On the Management of Multidimensional Discrete Data. VLDB Journal 4(3)1994, Special Issue on Spatial Database Systems, pp. 401 – 444.

[3] Baumann, P., 1999. A Database Array Algebra for Spatio-Temporal Data and Beyond. Proc. 4th Intl. Workshop on Next Generation Information Technologies and Systems (NGITS '99), Zikhron Yaakov, Israel, Springer LNCS 1649.

[4] Baumann, P., 2009. Web Coverage Processing Service (WCPS) Standard. Version 1.0, OGC 08-068r2.

[5] Baumann, P., 2010. The OGC Web Coverage Processing Service (WCPS) Standard. Geoinformatica, 14(4)2010, pp 447-479.

[6] Baumann, P., 2012. OGC GML 3.2.1 Application Schema – Coverages. Version 1.0, OGC 09-146r2.

[7] Baumann, P., 2012. OGC WCS Interface Standard – Core. version 2.0.1, OGC 09-110r4.

[8] Baumann, P., Hirschorn, E., Maso, J., 2016. OGC Coverage Implementation Schema. Version 1.1, OGC 09-146r3.

[9] Baumann, P., Mazzetti, P., Ungar, J., Barbera, R., Barboni, D., Beccati, A., Bigagli, L., Boldrini, E., Bruno,

R.. Calanducci, A., Campalani, P., Clement, O., Dumitru, A., Grant, M., Herzig, P., Kakaletris, G., Laxton, J., Koltsida, P., Lipskoch, K., Mahdiraji, A.R., Mantovani, S., Merticariu, V., Messina, A., Misev, D., Natali, S., Nativi, S., Oosthoek, J., Passmore, J., Pappalardo, M. Rossi, A.P., Rundo, F., Sen, M., Sorbera, V., Sullivan, D., Torrisi, M., Trovato, L., Veratelli, M.G., Wagner, S., 2015. Big Data Analytics for Earth Sciences: the EarthServer Approach. Intl. Journal of Digital Earth, 0(0)2015, pp. 1 – 27

[10] Baumann, P., Rossi, A.P., Misev, D., Dumitru, A., Merticariu, V., Huu, B.P., Figuera, R.M., Kakaletris, G., Koltsida, P., Siemen, S., Wagemann, J., Clements, O., Hogan, P., 2016: Big Earth Data at Your Fingertips. In: P.P. Mathieu, C. Aviset (eds.): *Earth Observation Open Science and Innovation*, ISSI Publications (to appear).

[11] Botts, M., Robin, A., 2014. OGC® SensorML: Model and XML Encoding Standard. Version 2.0, OGC 12-000.

[12] Broering, A., Stasch, C., Echterhoff, J., 2012. ®OGC Sensor Observation Service Interface Standard. Version 2.0, OGC 12-006.

[13] Cantini, F., Rossi, A.P., Orosei, R., Baumann. P., Misev, D., Oosthoek, J., Beccati, A., Campalani, P., Unnithan, V., 2014. MARSIS Data and Simulation Exploited Using Array Databases: PlanetServer/EarthServer for Sounding Radars. Geophysical Research Abstracts, Vol. 16, EGU2014-3784.

[14] Colaiacomo, L., Maso, J., Devys, E., 2013. OGC® GML in JPEG 2000 (GMLJP2) Encoding Standard Part 1: Core - with Corrigendum. Version 2.0.1, OGC 08-085r5.

[15] Dumitru, A., Merticariu, V., Baumann, P., 2014. Exploring Cloud Opportunities from an Array Database Perspective. Proc ACM SIGMOD Workshop on Data Analytics in the Cloud (DanaC'2014), Snowbird, USA.

[16] EarthServer, 2016. EarthServer: Big Datacubes at Your Fingertips. www.earthserver.eu, seen 2016-05-05.

[17] EarthServer, 2016. Marine Data Service, http://earthserver.pml.ac.uk, 2016-05-05.

[18] Google, 2016. Google Web Toolkit. http://www.gwtproject.org/, seen 2016-05-05.

[19] Gudgin, M., et al, 2007. SOAP. Version 1.2, https://www.w3.org/TR/soap12/, seen 2016-05-05.

[20] ISO, 2005. 19123:2005 Geographic information -- Schema for coverage geometry and functions.

[21] Liakos, P., Koltsida, P., Baumann, P., Ioannidis, Y., Delis, A., 2015. A Distributed Infrastructure for Earth-Science Big Data Retrieval. Intl. Journal of Cooperative Information Systems, 24(2)2015.

[22] Merticariu, G., Misev, D., Baumann, P., 2015. Measuring Storage Access Performance in Array Databases. Proc. 7th Workshop on Big Data Benchmarking (WBDB), New Delhi, India.

[23] Misev, D., Baumann, P., 2015. Enhancing Science Support in SQL. Proc. Workshop Data and Computational Science Technologies for Earth Science Research (co-located with IEEE Big Data), Santa Clara, US.

[24] Misev, D., Rusu, M., Baumann, P., 2012. A Semantic Resolver for Coordinate Reference Systems. Proc. 11th International Symposium on Web and Wireless Geographical Information Systems (W2GIS), Naples, Italy, Springer LNCS 7236.

[25] Mueller, M., Pross, B., 2014. OGC® WPS 2.0 Interface Standard Corrigendum 1. Version 2.0.1, OGC 14-065.

[26] OGC, 2011. The OpenGIS Abstract Specification – Topic 6: Schema for coverage geometry and functions. OGC 07-011.

[27] OGC, 2016, WCS 2.0 (OGC 09-110r4) compliance test suite.http://cite.opengeospatial.org/teamengine/about/wcs/2.0.1/site, seen 2016-05-05.

[28] OMG, 2012. CORBA. Version 3.3, http://www.omg.org/spec/CORBA/3.3/, seen 2016-05-05.

[29] PlanetServer, 2016. Planetary Science Data Service. www.planetserver.eu, 2016-05-05.

[30] Portele, C., 2007. OpenGIS® Geography Markup Language (GML) Encoding Standard. Version 3.2.1, OGC 07-036.

[31] Portele, C., 2012. OGC® Geography Markup Language (GML) — Extended schemas and encoding rules. Version 3.3, OGC 10-129r1

[32] Robin, A., 2011. OGC ®OGC SWE Common Data Model Encoding Standard. Version 2.0, OGC 08-094r1

[33] Tennison, J., 2014. The RDF Data Cube Vocabulary. W3C Recommendation, https://www.w3.org/TR/vocab-data-cube, seen 2016-05-05.

[34] Tomlin, C.D., 1990. A map algebra. Harvard Graduate School of Design.

[35] Wikipedia, 2016. Web Coverage Service. https://en.wikipedia.org/wiki/Web_Coverage_Service, seen 2016-05-05.

[36] Wikipedia. Java Remote Method Invocation. https://en.wikipedia.org/wiki/Java_remote_method_invocation, seen 2016-05-05.