Component-based Web Clients For Scientific Data Exploration Using The DCC Framework

A. Santanchè[1], P. Baumann[2]

[1] UNIFACS (Salvador) and UNICAMP (Campinas), Brazil
Email: santanche@ic.unicamp.br

[2]Jacobs University, Campus Ring 12, 28759 Bremen, Germany
Email: p.baumann@jacobs-university.de

1. Motivation

In this age we perceive an exploding number of services for online data exploration and exploitation in all scientific domains, such as weather observation (Adaguc 2010), planetary sciences (MapAPlanet 2010), astrophysics (Sloan 2010), earth sciences (Pangaea 2010), biology (MegDB 2010), human brain imaging (SenseLab 2010), and gene data (GenBank 2010), to name but a few. While high emphasis is put on modular, open, and standards-based service interfaces, there is lesser emphasis on methodologies for constructing clients with similar properties.

A closer look at client internals reveals that usual techniques to produce Web-based front-ends still are document-centric. The script implementation still is a largely manual, often painful task. In our view there is a lack in design methodologies and tools for a component-oriented Web GUI design of scientific information services. There is a need for clients to seamlessly adapt to a broad variety of specialized domains, tasks, and user preferences. As desktop GUI development teaches us, such an approach can offer several benefits: reduced client development cycles, pluggable components, improved unit testing, and ultimately cleaner code which is less error prone and better to maintain.

Our approach follows a component-based approach where interfaces can be interactively built and tested by a direct plugging and manipulation of components. We present our ongoing work on a component-driven approach to produce interactive Web interfaces for scientific data exploration, based on the Digital Content Component (DCC) by Santanche and Medeiros (2007).

2. Related Work

2.1 The DCC Framework

Digital Content Component (DCC), (Santanchè and Medeiros 2007) is a component model and framework which has the following main characteristics: (i) **interoperability** by adopting open standards for components and their interface descriptions, comprising XML, OWL and WSDL; (ii) a **layered communication protocol** combining performance (lower layers) and ease of use plus interoperability (upper layers); (iii) **homogeneity** – the same model affords units containing executable software (Process DCCs) and units containing non-executable content represented as complex digital objects (Passive DCCs).

The division between Process and Passive DCCs enables to distinctly handle them and connect both at will. Through a mechanism we call *content-driven execution* a Passive DCC is connected to a Process DCC containing executable software enabled to handle its content. A specialization of the Passive DCC is the Bridge DCC (Pastorello Jr et al. 2008), which works as a

bridge between a remote data source and a local application. In this work we apply Bridge DCCs to fetch remote geo data.

2.2 The OGC WCPS Geo Raster Language Standard

The Open GeoSpatial Consortium (OGC) develops and maintains open, interoperable geo service standards. One member of the family of modular standards is the Web Coverage Processing Service (WCPS) Language Interface Standard (Baumann 2008), which defines a query language for retrieval on multi-dimensional raster data sets. Generally speaking, the language is suitable for navigation, visualization, extraction, and analysis of sensor, image, and statistics data. By nesting expressions, tasks of unlimited complexity can be formulated. WCPS, therefore, has been dubbed "SQL for coverages". We briefly introduce WCPS; see (Baumann 2009) for details.

Model and Language

In OGC, a *coverage* is defined as a function from some spatio-temporal domain to values of some range set (Baumann 2010). WCPS operates on a practically important category of coverages, namely *quadrilateral grid coverages*, commonly also known as *raster data*. On such coverage objects, WCPS requests extract either new coverages or summary data. We give a flavour of the language by way of an example. The following request inspects three coverages *Modis1*, *Modis2*, and *Modis3* in turn, picks those where average red channel intensity exceeds 127, and delivers the difference between red and near-infrared (*nir*) band, encoded in TIFF:

```
for $m in ( Modis1, Modis2, Modis3 )
where
    avg( $m.red ) > 127
return
    encode( abs( $m.red - $m.nir ), "tiff" )
```

The `for` part defines an iteration variable which successively is bound to each of the coverages addressed in parentheses; such “loops” can be nested, thereby allowing to combine several coverages in one request, e.g., for sensor fusion. The optional `where` clause allows filtering out unwanted coverages. Each coverage passing the filter gets evaluated in the `return` clause. In our case, this is the absolute of the difference between the two channels, applied to all coverage pixels simultaneously. In case of coverage-valued results as is the case here, the result needs to be encoded in some suitable data format, which is accomplished by the `encode()` function.

Implementation

The open-source WCPS reference implementation, petascope (see www.petascope.org), allows clients to submit query requests for evaluation by the server, which responds with a result set. Petascope is the OGC layer of the rasdaman array DBMS (see www.rasdaman.org) which allows querying raster data stored in a PostgreSQL database. Extensive storage and query optimization is applied to incoming requests, such as adaptive tiling and compression, algebraic rewriting, just-in-time compilation, dynamic pre-aggregation, parallel and distributed query processing, and several more.

3. Towards A Component-Oriented Client Design Methodology

Departing from the DCC platform running in the Web browser, we designed an architecture following the rationale of the Model-View-Control (Krasner 1988) architectural style.

Proxy DCCs – Simplify the communication between client and server through a class of components which mirror the server interface in the client side, dispatching the requests to the server and receiving the remote responses, delivering them to the requester. A proxy DCC takes the Model role.

View DCCs – Enable a visual presentation and exploration of data.

Control DCCs – Mediate the relationship among other DCCs and interact with the user, dispatching requests to other components when necessary.

Figure 1 summarizes our approach to produce component-based scientific applications. It works in three steps: (1) a Web browser retrieves an HTML page embedding a scientific application composition; (2) JavaScript components referred to in the composition are retrieved from the DCC repository, dynamically instantiated, and combined in the client browser; (3) application starts to run and a WCPS Proxy DCC which connects to a WCPS server; data fetched which will be processed by the application as detailed below.

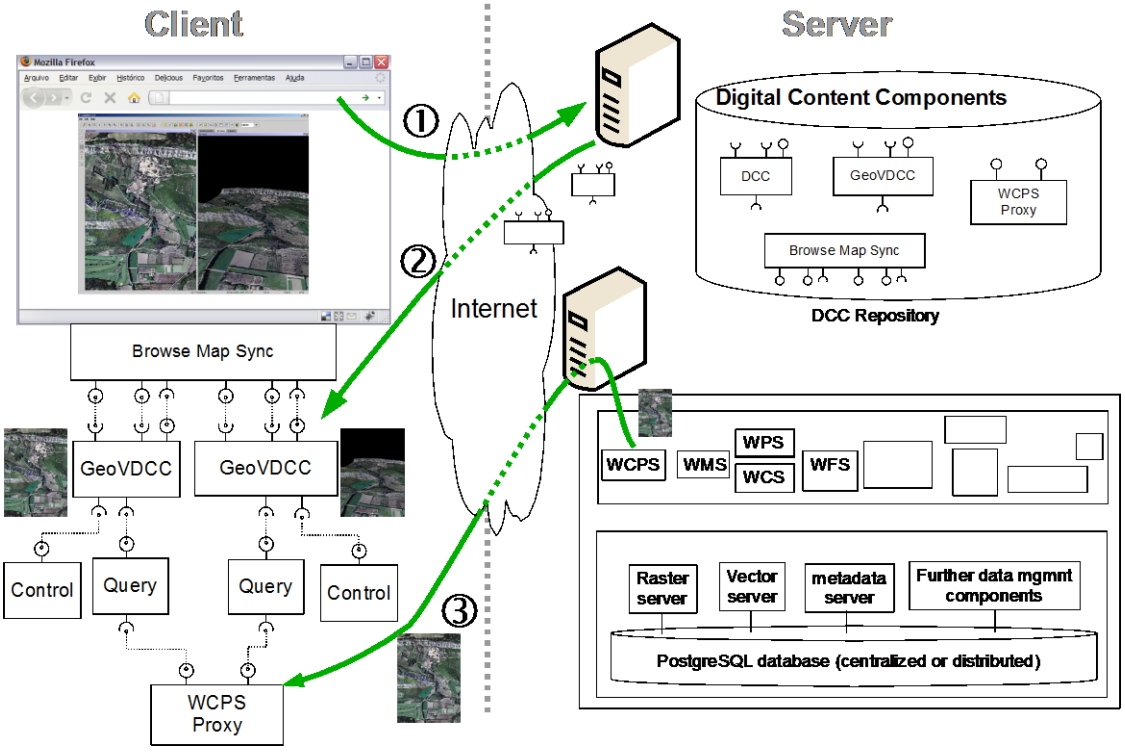

Figure 1. Example of DCC composition to present two synchronized maps.

A scientific application is formed by a composition which is described as a set of DCC instances connected by their interfaces. Instances are tailored through the specification of property values.

The strategy to embed these applications within HTML pages is an important contribution of our approach. Instead of recurring to complex JavaScript programming, our solution adopts the Microformats (`http://microformats.org`) standard in order to semantically enrich HTML documents with metadata describing DCC instances, their properties, and connections. The Microformats approach adopts elements previously existing in the HTML standard which are

repurposed to express new meanings. Consider the following example showing a snippet of an application embedded in an HTML page:

```
<div class="dcc" id="myquery">
    <div class="property" name="type">http://purl.org/dcc/query</div>
</div>
<div class="connector" id="query2proxy">
    <div class="origin">myframe</div>
    <div class="destination">myquery</div>
</div>
```

`<div>` tags demark DCC instances and connections. The `class` attribute specifies the role of each `<div>`. In the example, there is a `<div>` specifying a DCC instance and a second `<div>` specifying a connection between two DCCs. Inner `<div>`s are meant to describe component and/or connection characteristics – e.g., component properties, connection origin, and destination.

Experiments

In a first practical use case scenario we have implemented a sample service allowing to retrieve vegetation information from a multi-spectral Landsat image. In remote sensing, the Normalized Difference Vegetation Index (NDVI) is widely used to determine areas covered by live green vegetation. For a multi-spectral coverage with channels `red` and `nir` (near-infrared) it is defined as

```
NDVI = (c.nir - c.red) / (c.nir + c.red)
```

Values range between –1 and +1, whereby values close to +1 indicate photosynthetic activities.

Our scenario displays the original image and a thresholded NDVI image in which pixel values below the threshold are mapped to black and those above to white. An interaction panel allows adjusting the currently used threshold value; upon every change, the NDVI is dynamically fetched from the database. The corresponding WCPS template is given as

```
for $s in ( LandsatScene )
 return
    encode( (($s.nir-$s.red)/($s.nir + $s.red) > #t#)*255, "tiff")
```

The `#t#` marker indicates the threshold value to be substituted by the DCC in charge.

Three interaction DCCs – two buttons and an input field – control the visual DCC, which in turn makes use of the server connectivity DCC for fetching an image whenever requested. Fig. 2 shows the corresponding web page appearance.

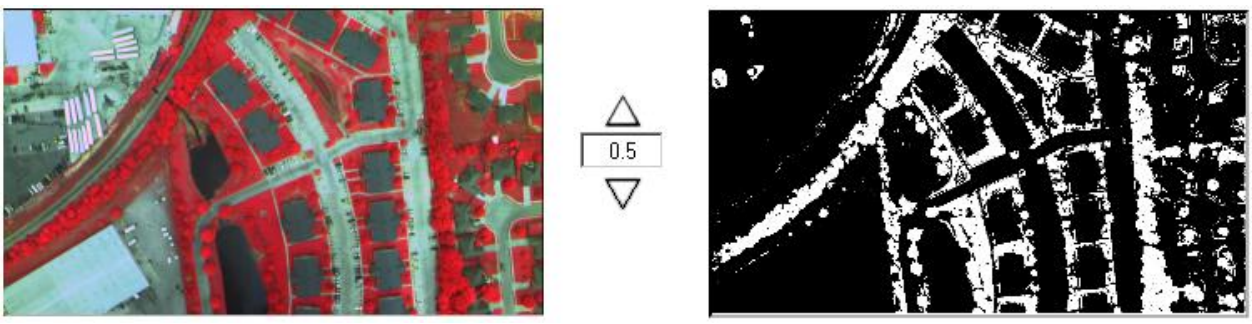

Figure 2: Screenshot of the NDVI retrieval use case.

4. Conclusion and Outlook

We feel that building Web services in a component-oriented manner, on both client and server, results in increased flexibility, customizability, and overall software quality.

Our example shown is but a first step. Next, we plan to explore more complex client scenarios where several interaction components are coupled with a number of display components in m:n relationships. Based on these experiences, the conceptual fundament of the DCC framework will be completed. Further, it is foreseen to completely hide the JavaScript code in HTML; this is expected to ease automatic client generation. Ultimately, we envisage toolkits which combine the flexibility of WCPS-style request languages and component-based GUIs to allow generating bespoke data exploration clients on demand.

Acknowledgments

This work was developed within a binational project financed by CNPq (Brazil) and BMBF (Germany). It was also partially financed by CNPq (BioCORE project), INCT in Web Science (CNPq 557.128/2009-9), FAPESP, CAPES and FAPESB (LabMultiflex project), as well as a project grant from CNPq.

References

Adaguc, http://adaguc.knmi.nl/. Last seen April 16, 2010

Baumann, P., 2009, The OGC Web Coverage Processing Service (WCPS) Standard, *Geoinformatica*, http://dx.doi.org/10.1007/s10707-009-0087-2

Baumann, P. (ed.), 2008, Web Coverage Processing Service (WCPS) Language Interface Specification. OGC 08-068r2

Baumann, P., 2006, Large-Scale Raster Services: A Case for Databases (invited keynote). In: John Roddick et al (eds): *Advances in Conceptual Modeling - Theory and Practice*, Proc. CoMoGIS, Tucson, USA, 6 - 9 November 2006, LNCS 4231, Springer 2006, 75 – 84

Baumann, P. (ed.), 2010, "GML 3.2.1 Application Schema for Coverages", OGC 09-146r1

GenBank, http://www.ncbi.nlm.nih.gov/Genbank/. Last seen April 16, 2010

GeoSciML, http://www.geosciml.org/. Last seen April 16, 2010

Krasner1988, G. Krasner, S. Pope, 1988, A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 system, *Journal of Object Oriented Programming*, 1(3):26–49

MapAPlanet, http://www.mapaplanet.org/. Last seen April 16, 2010

MegDB, http://www.megx.net/. Last seen April 16, 2009

Pangaea, http://www.pangaea.de/. Last seen April 16, 2010

Pastorello Jr, GZ, Medeiros, CB, and Santanchè, A, 2008. Accessing and processing sensing data. In: *CSE '08: Proc. of the 2008 11th IEEE Int. Conf. on Computational Science and Engineering*, Washington, DC, USA, 353–360.

Santanchè, A and Medeiros, CB, 2007, A Component Model and Infrastructure for a Fluid Web. *IEEE Trans. on Knowledge and Data Engineering*, 19(2):324–341.

Senselab, http://senselab.med.yale.edu/. Last seen April 16, 2010

Sloan, http://www.sdss.org/. Last seen April 16, 2010